400069 Rev. 1

National Semiconductor

Microcontrollers Databook

A Corporate Dedication to Quality and Reliability

National Semiconductor is an industry leader in the manufacture of high quality, high reliability integrated circuits. We have been the leading proponent of driving down IC defects and extending product lifetimes. From raw material through product design, manufacturing and shipping, our quality and reliability is second to none.

We are proud of our success . . . it sets a standard for others to achieve. Yet, our quest for perfection is ongoing so that you, our customer, can continue to rely on National Semiconductor Corporation to produce high quality products for your design systems.

Charles E. Sporck
President, Chief Executive Officer
National Semiconductor Corporation

Wir fühlen uns zu Qualität und Zuverlässigkeit verpflichtet

National Semiconductor Corporation ist führend bei der Herstellung von integrierten Schaltungen hoher Qualität und hoher Zuverlässigkeit. National Semiconductor war schon immer Vorreiter, wenn es galt, die Zahl von IC Ausfällen zu verringern und die Lebensdauern von Produkten zu verbessern. Vom Rohmaterial über Entwurf und Herstellung bis zur Auslieferung, die Qualität und die Zuverlässigkeit der Produkte von National Semiconductor sind unübertroffen.

Wir sind stolz auf unseren Erfolg, der Standards setzt, die für andere erstrebenswert sind. Auch ihre Ansprüche steigen ständig. Sie als unser Kunde können sich auch weiterhin auf National Semiconductor verlassen.

La Qualité et La Fiabilité:

Une Vocation Commune Chez National Semiconductor Corporation

National Semiconductor Corporation est un des leaders industriels qui fabrique des circuits intégrés d'une très grande qualité et d'une fiabilité exceptionelle. National a été le premier à vouloir faire chuter le nombre de circuits intégrés défectueux et a augmenter la durée de vie des produits. Depuis les matières premières, en passant par la conception du produit sa fabrication et son expédition, partout la qualité et la fiabilité chez National sont sans équivalents.

Nous sommes fiers de notre succès et le standard ainsi défini devrait devenir l'objectif à atteindre par les autres sociétés. Et nous continuons à vouloir faire progresser notre recherche de la perfection; il en résulte que vous, qui êtes notre client, pouvez toujours faire confiance à National Semiconductor Corporation, en produisant des systèmes d'une très grande qualité standard.

Un Impegno Societario di Qualità e Affidabilità

National Semiconductor Corporation è un'industria al vertice nella costruzione di circuiti integrati di altà qualità ed affidabilità. National è stata il principale promotore per l'abbattimento della difettosità dei circuiti integrati e per l'allungamento della vita dei prodotti. Dal materiale grezzo attraverso tutte le fasi di progettazione, costruzione e spedizione, la qualità e affidabilità National non è seconda a nessuno.

Noi siamo orgogliosi del nostro successo che fissa per gli altri un traguardo da raggiungere. Il nostro desiderio di perfezione è d'altra parte illimitato e pertanto tu, nostro cliente, puoi continuare ad affidarti a National Semiconductor Corporation per la produzione dei tuoi sistemi con elevati livelli di qualità.

Charles E. Sporck
President, Chief Executive Officer
National Semiconductor Corporation

MICROCONTROLLER
DATABOOK

1988 Edition

TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

AbuseableTM
AnadigTM
ANS-R-TRANTM
APPSTM
Auto-Chem DeflasherTM
BCPTM
BI-FETTM
BI-FET IITM
BI-LINETM
BIPLANTM
BLCTM
BLXTM
Brite-LiteTM
BTLTM
CheckTrackTM
CIMTM
CIMBUSTM
Clock✓ChekTM
COMBOTM
COMBO ITM
COMBO IITM
COPSTM microcontrollers
DatacheckerTM
DENSPAKTM
DIBTM
Digitalker®
DISCERNTM
DISTILLTM
DNR®
DPVMTM
ELSTARTM
E-Z-LINKTM
FACTTM
FASTTM
5-Star ServiceTM
GAL®
GENIXTM
GNXTM
HEX 3000TM
HPCTM
ICMTM
INFOCHEXTM
Integral ISETM
IntelisplayTM
ISETM
ISE/06TM
ISE/08TM
ISE/16TM
ISE32TM
KeyScanTM
LMCMOSTM
M2CMOSTM
MacrobusTM
MacrocomponentTM
Meat✓ChekTM
Microbus™ data bus
MICRO-DACTM
μtalkerTM
MicrotalkerTM
MICROWIRETM
MICROWIRE/PLUSTM
MOLETM
MSTTM
Naked-8TM
National®
NAX 800TM
Nitride PlusTM
Nitride Plus OxideTM
NMLTM
NOBUSTM
NSC800TM
NSX-16TM
NS-XC-16TM
NURAMTM
OXISSTM
P2CMOSTM
Perfect WatchTM
Pharma✓ChekTM
PLANTM
PMPTM
PolycraftTM
POSitalkerTM
Power & ControlTM
QUAD3000TM
QUIKLOOKTM
RATTM
RTX16TM
SABRTM
Script✓ChekTM
SCXTM
SERIES/800TM
Series 3000TM
Series 32000®
Shelf✓ChekTM
SPIRETM
STARTM
StarlinkTM
STARPLEXTM
STARPLEX IITM
SuperChipTM
SuperScriptTM
SYS32TM
TapePak®
TDSTM
TeleGateTM
The National Anthem®
Time✓ChekTM
TINATM
TLCTM
TrapezoidalTM
TRI-CODETM
TRI-POLYTM
TRI-SAFETM
TRI-STATE®
TURBOTRANSCEIVERTM
VIPTM
VR32TM
WATCHDOGTM
XMOSTM
XPUTM
Z STARTM
883B/RETSTM
883S/RETSTM

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, California 95052-8090 (408) 721-5000
TWX (910) 339-9240

National Semiconductor

Microcontroller Introduction

Practical Solutions to Real Problems

Microcontrollers have always been driven by customer need rather than technological capability.

They were designed to meet specific needs with specific performance in specific applications with specific cost.

That also meant, however, that your choices were limited to what was available on the market—which meant possibly having to compromise your design objectives because you couldn't get exactly the microcontroller you needed.

No more.

Now you can get a microcontroller from National that spans a wide range of system solutions—to go almost anywhere your design imagination takes you.

Whether you need a low-cost 4-bit workhorse or a 16-bit 30 MHz powerhouse, whether you want ½ kbyte of ROM or over 64 kbytes, whether you're building a simple singing greeting card or a complex telecommunications network, we have a microcontroller for the job.

With on-board CPU, memory, internal logic, and I/Os, National microcontrollers are helping more and more designers lower system costs and shrink system size.

And as technology brings more peripheral functions onto the chip, including user-programmable memory, fast SRAM, timers, UARTs, comparators, A/D converters, and LAN interfaces, the microcontroller will become the cost-efficient choice for even such real-time "microprocessor" applications as laser printers, ISDN, and digital signal processing.

That's why National continues to lead the industry in the development of microcontroller technology.

That's why we're including our 8-bit and 16-bit controller cores in our standard-cell library.

That's why we're scaling our common M²CMOS™ process for submicron feature sizes, hypermegahertz frequencies, and unparalleled performance levels.

That's why we offer you "Hot-Line" applications support and a 24-hour-a-day digital information service.

That's why we offer you IBM®-PC and DEC™- VAX™-based development tools and high-level-language (C) compilers

And that's why we've committed the full resources of our company to provide you with the most complete, most reliable, most cost-effective systems solution for all your needs.

This databook is a reflection of that committment.

It will give you an overview of microcontrollers in general and of National's microcontrollers in particular.

It will help you evaluate your microcontroller options from both a business perspective and an engineering perspective.

It will help you make reasoned judgements about selecting the best microcontroller for your needs.

And it will show you what the microcontroller future holds in store for all of us.

If you'd like more information, or you'd like to find out how to put a microcontroller to work in your own application, just contact your local National Semiconductor Sales Office.

National Semiconductor

How to Select a Microcontroller

Microcontrollers have evolved far beyond their origins as control chips in calculators.

Today, microcontrollers can be the perfect solution for simplifying a wide range of designs. And for giving those designs a clear competitive advantage in the marketplace.

Whether used for simple logic replacement or as an integral part of a high performance system, a microcontroller can reduce system costs, shrink system size, and shorten system design cycles. And yet deliver performance often superior to "traditional" digital solutions.

Still, all microcontrollers are not created equal. And it's important to consider a number of factors before committing to a particular device:

1. Is the microcontroller optimized for your specific application in terms of speed, performance, features, and cost?
2. Is it code-efficient, and based on a true microcontroller architecture for the highest performance and efficiency?
3. Is it fabricated in the most advanced CMOS process technology, and is it fully scalable to maintain its performance edge in the future?
4. Is it supported by a comprehensive family of development tools that run on standard platforms such as the IBM-PC and DEC VAX?
5. Is it backed by a dedicated team of professionals who are available not only to provide expert training for new users, to get them on-line quickly and efficiently, but also to provide technical guidance for even the most experienced user?
6. Is it designed for the future, with the capability of on-chip gate arrays and with the planned implementation of the controller core as a standard-cell functional block?

If you answered "yes" to all these questions, then you already know that there's only one company with the product depth and technology capability to provide you with a microcontroller optimized for your specific application.

National Semiconductor.

You'll find National Microcontrollers in:

Laser Printers
Disc Controllers
Telecommunications Systems
Keyboards
Airplane Multiplex Systems
Car Radios
Engine Control Systems
Anti-Skid Brake Systems
Armaments
Factory Automation
Medical Equipment
Fuses
Scales
Refrigerators
Security Systems
Garage Door Openers
Camera Aperture Controls
Office Copiers
Cable TV Converters
Televisions
Video Recorders
Solar Heating Controls
Thermostats
Climate Control Systems
Intelligent Toys
Kitchen Timers

National Semiconductor

Why Select a National Microcontroller

National has created the most complete selection of 4-, 8-, and 16-bit microcontrollers of any company in the industry.

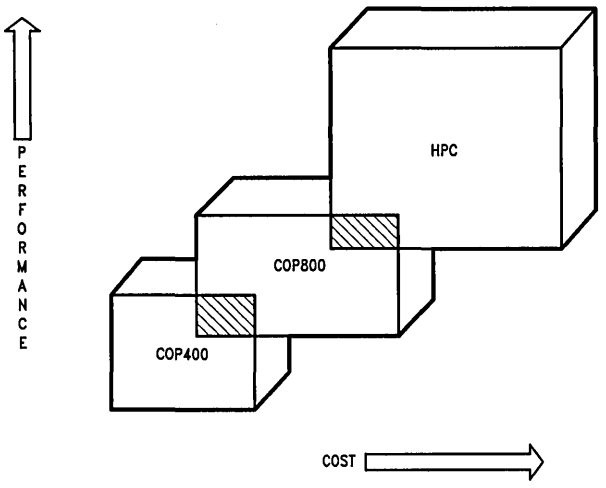
Which means that no matter what the specific needs of your application are, you can find a National microcontroller to meet them.

Our COP400 family offers the lowest-cost, 4-bit solutions for timing, counting, and control functions.

Our COP800 family offers low-cost, feature-rich, 8-bit solutions.

And our High Performance microController (HPC™) family offers the highest performance with the world's fastest 16-bit CMOS solution.

Microcontroller Family of Products



TL/XX/0071–1

With a full range of performance- and feature-options, National's microcontroller families can be customized to meet the needs of your specific application.

1.0 COMMON FEATURES FOR A CUSTOM FIT

All our microcontrollers are designed to provide not just a one-time-only solution, but a continuum of solutions to meet the changing demands of your product and the marketplace.

Our COP400 family, for example, which consists of over 60 devices, is designed with a common instruction set, so you can migrate from one member of the family to others without having to recode, so you can take efficient advantage of the application-specific flexibility of the COP400 family's programmable I/O options.

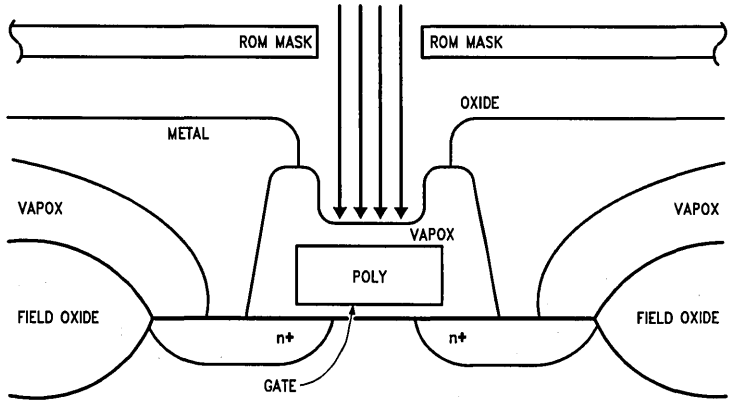
Our COP800 and HPC families, on the other hand, are each designed around a common CPU core that then can be surrounded by a variety of standard functional building blocks such as RAM, ROM, user programmable memory, fast SRAM, DMA, UART, comparator, A/D, HDLC, and I/O.

This unique core approach allows us to offer you a microcontroller with the exact combination of CPU power and peripheral function you need for your specific application. So you don't have to compromise your design parameters by using an inappropriate device, and you don't have to compromise your cost parameters by paying for performance and features you don't need.

This core concept also allows us to bring new microcontroller products to market fast and at a lower cost to help you keep pace with the rapidly changing conditions in your own market.

And it allows us to implement both the COP800 and the HPC cores as standard cells, for the highest levels of integration and flexibility in your own proprietary design.

COP400 Post-Metal ION Implant



TL/XX/0071–2

2.0 TRUE MICROCONTROLLER ARCHITECTURE

Our microcontrollers are designed as true controllers, not modified microprocessors.

The COP400 family is designed with a two-bus Harvard architecture; the COP800 family with a memory-mapped, modified Harvard architecture, and the HPC family with a memory-mapped, von Neumann architecture.

All three control-oriented families, however, are optimized for high code efficiency. Most instructions are only 1 byte long—yet each can typically execute several functions. This "function-dense" code provides a substantial increase in memory efficiency and processing speed.

3.0 ADVANCED PROCESS AND PACKAGING TECHNOLOGIES

National offers you not only the right microcontroller for your needs, but also the right process technology for your microcontroller.

COP400 devices are available in both high-speed NMOS and low-power CMOS fabrications, while the higher-performance COP800 and HPC families are both fabricated in National's advanced M²CMOS process.

M²CMOS. This double-metal CMOS process offers significant design advantages. It combines the speed of NMOS, the ruggedness of bipolar, and the low power consumption of bulk CMOS to produce fast, dense, highly efficient, highly scalable devices for a wide variety of integrated-circuit designs.

It's for these reasons that M²CMOS has become the standard process technology for all of National's advanced-technology LSI and VLSI products, including microprocessors, gate arrays, standard cells, telecommunications devices, linear devices and, of course, microcontrollers.

Post-Metal Programming (PMP). This is a new process technology available from no other semiconductor manufacturer in the world. It offers the fastest, guaranteed prototype programmed-ROM turn-time in the industry.

PMP is a high-energy implantation process that allows microcontroller ROM to be programmed **after** final metallization.

This is a true innovation, because ROM is usually implemented in the second die layer, with nine or ten other layers then added on top. And that means the ROM pattern must be specified early in the production process, and completed prototype devices won't be available typically for six weeks.

With PMP, however, dice can be fully manufactured through metallization and electrical tests (only the passivation layers need to be added), and held in inventory. Which means ROM can be programmed late in the production cycle, **making prototypes available in only two weeks!**

And production parts can follow in as little as four weeks.

PMP allows you to adapt to fast-changing market conditions and to take maximum advantage of narrow windows of opportunity.

And shorter production lead times can simplify your inventory control and reduce safety stock by up to 20%, giving you significant cost reductions.

Currently, Post-Metal Programming is available for selected members of the COP400 family, and will be expanded to the COP800 and HPC familes in the near future.

Military versions. All National microcontrollers have CMOS parts available in the full military temperature range (−55°C to +125°C).

In addition, parts are available that have been certified under MIL-STD-883, Rev. C, the most rigorous non-JAN screening flow in the electronics industry.

Packaging. One major reason that National microcontrollers demonstrate such consistently high levels of reliability is that we've developed special advanced packaging processes to protect the die.

For example, we've designed a unique leadframe with "locking holes" that helps block any penetrating moisture from reaching the die itself.

And the leadframes themselves are made of an unusual high-strength copper alloy that has a lower thermal resistance (θ_{JA}) than typical Alloy 42-leadframes.

We've also employed a unique low-stress, high-purity epoxy molding compound for our packages, which gives them a coefficient of expansion that nearly matches that of the leadframes. As a result, many of our microcontrollers are also offered in plastic packages for military-temperature-range operation.

Reliability is built-in at the die level as well. Our M²CMOS microcontrollers are fabricated on dedicated lines at our world-class, six-inch wafer-fab facility in Arlington, Texas. With its Class-10 clean rooms and automated-handling system, Arlington has set a standard of reliability equalled by few other companies in the industry.

And this reliability is available to you in a wide variety of microcontroller packages, ranging in size from 20 to 84 pins.

Package types include plastic and ceramic DIPs, small outline (S.O.) surface mounts, plastic and ceramic leaded chip carriers, and pin grid arrays.

Or, you can select the world's most advanced, high-density packaging option, TapePak™.

TapePak comines the advantages of an automated tape-and-reel-type delivery system with built-in testing pads for reliability and a unique plastic package carrier. The result is a surface-mounted package that can be as small as $\frac{1}{10}$ the size of conventional surface mounts, with lead spacings of 20 mils.

4.0 FULL DEVELOPMENT SUPPORT

Even the right microcontroller, of course, is useless without the right development tool to put that controller to work in your application.

That's why National offers you a full range of development support. Ready-to-run evaluation boards. Emulators. Software. Prototyping devices. Training and seminars for beginning and advanced users. Everything you need to take your design from concept to reality.

And you don't need an expensive development environment to do it. With our exclusive Microcontroller On-Line Emulator (MOLE™), a standard IBM PC or DEC VAX becomes a full-featured platform.

And with our comprehensive library of prewritten routines, from keyboard scanners to Fast Fourier Transforms, you can reduce software programming to a minimum. This "user-friendly" service can help you bring your design to market quickly and cost-effectively.

5.0 FULL APPLICATIONS SUPPORT

At National, we believe that applications support should be immediate and "hands-on".

That's why we established the unique Dial-A-Helper program.

With a computer, modem, and telephone, you can tie directly into our Microcontroller Applications Group for fast, direct assistance in developing your design.

You can leave messages on our electronic bulletin board for our Applications Engineers, who will respond to you directly.

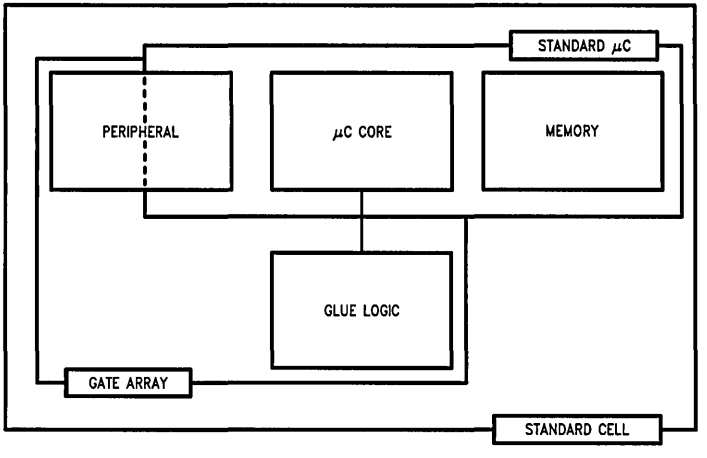
You can access applications files.

You can download those files for later reference.

Or, if you're having a real problem, you can actually turn the control of your Microcontroller On-Line Emulator development system over to our engineering staff, who can perform remote diagnostic routines to locate and eliminate any bugs.

The point is, when you buy a microcontroller from National, you're buying more than silicon—you're buying the commitment of an entire company of dedicated professionals who share a single goal: to help you put that silicon to **work.**

Systems In the Future—Integration Path



TL/XX/0071–3

6.0 THE ASIC FUTURE

National's microcontrollers were designed to meet two objectives: to adapt to your evolving needs, and to adapt to evolving technology.

Both "evolutions," however, are leading to the same goal: the complete "system-on-chip" solution. Already, the glue logic that ties a microcontroller to its peripheral functions can be replaced with a gate array. And soon, all three functions (microcontroller "core", logic, and peripherals) will be available as a single standard-cell functional block.

The key to achieving this goal, of course, is a common, advanced, scalable process technology.

That's why both the COP800 and HPC families are fabricated in our high-performance double-metal CMOS process. This is a highly scalable technology that can accommodate die shrinks to submicron feature sizes, increasing performance and cutting power consumption with each step.

Moreover, because M^2CMOS is now the standard process technology for all new National LSI and VLSI devices, the COP800 and HPC cores will not only be available as part of our standard-cell library, but will also be able to support one of the broadest range of functional blocks available from any semiconductor manufacturer—all aligned on the same set of design rules.

So you can standardize your designs on just one or two core processors, and, as we introduce new technologies and functions, you can maintain that design knowledge base while taking advantage of these new, higher levels of functional integration.

And because National (and **only** National) gives you the option of using standard parts or designing your own customized solutions—both supported by common design tools and a common process—you can create highly competitive, highly secure, highly optimized solutions in minimal space at minimal cost in minimal time.

And that's the name of the game.

Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

Alpha-Numeric Index

Alpha-Numeric Index (Continued)

Alpha-Numeric Index (Continued)

Alpha-Numeric Index (Continued)

Section 1

COP400 Family

Section 1 Contents

National Semiconductor

The 4-Bit COP400 Family: Optimized for Low-Cost Control

National's COP400 family offers the broadest range of low-priced, 4-bit microcontrollers on the market.

Key Features

- High-performance 4-bit microcontroller
- 4 μs–16 μs instruction-cycle time
- ROM-efficient instruction set
- On-chip ROM from 0.5k to 2k
- On-chip RAM from 32 x 4 to 160 x 4
- More than 60 compatible devices in family
- Common pin-outs
- NMOS and P²CMOS™
- MICROWIRE™ serial interface
- Wide operating voltage range: +2.4V to +9V
- Military temp range available: −55°C to +125°C
- 20- to 28-pin packages (incl. 20-, 24-pin SO and 28-pin PLCC)

And far from being "old technology," 4-bit microcontrollers are meeting significant market needs in more applications than ever before. In fact, National shipped more than 40 million 4-bit devices last year alone. The reason for the continuing strength of the COP400 family is its versatility. You can select from over 60 different, compatible devices. You can select devices with unit costs **below 50 cents**—the lowest-priced microcontrollers in the world. You can select devices with a wide variety of ROM and RAM combinations, from 0.5k ROM and 32 x 4 RAM to 2k ROM and 160 x 4 RAM.

And every COP400 family member shares the same powerful, ROM-efficient instruction set and the same pin-out, so you can migrate between devices without re-engineering.

And like all of National's microcontrollers, the COP400 can be optimized to meet your specific application needs, with a variety of I/O options, pin-outs, and package types, from DIPs to SMDs.

COPS™ microcontrollers can be used to replace discrete logic in high-volume consumer products and low-volume industrial products allowing you to add features, miniaturize and reduce component count.

Key Applications

- Consumer electronics
- Automotive
- Industrial control
- Toys/games
- Telephones

Wide Acceptance

COPS wide acceptance comes from innovative products. National has built on this established family with continued and enhanced devices.

- The first under-a-dollar microcontroller led to a broader range of automotive and consumer applications.
- The first high-speed, low-power CMOS microcontrollers with 0.5k ROM provides design flexibility at low cost.
- The first microcontroller implementing MICROWIRE/PLUS™ allowing two-way communication across only three lines.
- The first under $.50 microcontroller providing excellent cost/performance benefits for applications impossible before.
- The first microcontroller implementing Post-Metal Programming (PMP™) for quick turns prototyping and production.

PMP

Post-Metal Programming (PMP), another NSC microcontroller first. Takes advantage of:

- Seasonal or volatile market demand
- Narrow windows of opportunity in highly competitive markets
- Simplified inventory control
- Reduced safety stock

Get all the advantages of custom-programmed microcontrollers with all the business advantages of low cost, quick-turn prototyping and production.

The secret is an entirely new process technology called Post-Metal Programming.

PMP (Continued)

INSIDE PMP

Post-Metal Programming is a high energy implantation process that allows the ROM layer of a microcontroller to be programmed after final metallization. That means every die layer can be fully fabricated, except for the passivation layers, and held in inventory. Then when you request a ROM pattern, a ROM implant mask is generated and the buried ROM layer is programmed with an ion beam.

The wafer is passivated and cut into dice which are then packaged on a quick-turn line.

So in only two weeks, you've got prototypes.

4-WEEK PRODUCTION QUANTITIES

Wafer fab accounts for the majority of prototyping and production time for integrated circuits.

With PMP, however, the dice are essentially complete and in inventory.

So we can take your approved prototypes right into full production in as little as four weeks.

WINNING THE TIME-TO-MARKET RACE

The electronics market won't wait for anyone. If your competitors make a move, you've got to respond now.

You can't wait around for proof-of-design prototypes. Even a week can make a difference between success or failure. Between gaining market share or losing it. Between staying ahead of the other guys or falling behind. With PMP, you can stretch that lead by *weeks*. In fact, if you compare the quick-turn PMP process to conventional prototype-and-production timetables, you'll see that *you can actually gain as much as 3½ months over your competitors!*

NO EXTRA COST

PMP is available at *no extra cost*.

That means, for example, that National's COP413L, the world's lowest-priced microcontroller at $.49 in quantity, is available in the PMP process for . . . $.49 in quantity.

Compare that with the traditional "alternative" for quick-turn prototyping of user-programmable ROM. EPROM and EEPROM can easily drive your unit costs up to as much as $6!

And when you consider the additional cost-savings of being able to reduce your safety stock in inventory, knowing you can get quick-turns in a few weeks, the PMP process and National Semiconductor microcontrollers not only make good *engineering* sense, they make good *business* sense.

System Solutions

The COP400 family provides a flexible, cost-effective system solutions to all applications requiring timing, counting, or control functions.

And, bottom line, if a 4-bit controller can do the job, why pay more?

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On-Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs, and the HPC™ family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

See AN-456 for more information.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP400	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COPS-PB1	Personality Board	COP400 Personality Board Users Manual	420408189-001
	MOLE-COPS-IBM	Assembler Software for IBM	COP400 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424409497-002 420040416-001
	424410284-001	Programmers Manual		424410284-001

COP400 Family of Microcontrollers

Commercial Temp Version 0°C to +70°C	Industrial Temp Version −40°C to +85°C	Military Temp Version −55°C to +125°C	Technology	Description				Features							Development Tools		Data Sheet Page
				Memory		I/O											
				ROM (Bytes)	RAM (Digits)	I/O Pins	Serial I/O	Interrupt	Stack	Time Base Counter	Micro Bus	Typ. 5V Operat. Power	Max Standby at 3.3V	Size (Pins)	ROMless Device	Piggyback	
COP413L*	COP313L		NMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401L-X13/R13		1-70
COP414L*	COP314L		NMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401LN		1-97
COP410L	COP310L		NMOS Low Power	0.5k	32	19	Yes	No	2 Level	No	No	15 mW	7.5 mW	24	COP401LN		1-52
COP411L	COP311L		NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401LN		1-52
COP413C	COP313C		CMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-83
COP413CH	COP313CH		CMOS Hi Speed	0.5k	32	15	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-83
COP410C	COP310C	COP210C (Note 1)	CMOS Hi Speed	0.5k	32	19	Yes	No	2 Level	No	No	1 mW	0.1 mW	24	COP404CN	COP444CP	1-37
COP411C	COP311C	COP211C (Note 1)	CMOS Hi Speed	0.5k	32	16	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-37
COP420	COP320		NMOS Hi Speed	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	100 mW	N/A mW	28	COP402N	COP420P	1-112
COP421	COP321		NMOS Hi Speed	1.0k	64	19	Yes	No	3 Level	Yes	No	100 mW	N/A mW	24	COP402N	COP420P	1-112
COP422	COP322		NMOS Hi Speed	1.0k	64	16	Yes	No	3 Level	Yes	No	100 mW	N/A mW	20	COP402N	COP420P	1-112
COP424C*	COP324C	COP224C (Note 2)	CMOS Hi Speed	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	28	COP404CN	COP444CP	1-161
COP425C*	COP325C	COP225C (Note 2)	CMOS Hi Speed	1.0k	64	19	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	24	COP404CN	COP444CP	1-161
COP426C*	COP326C	COP226C (Note 2)	CMOS Hi Speed	1.0k	64	16	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-161
COP420L*	COP320L		NMOS Low Power	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	45 mW	9.9 mW	28	COP404LSN-5	COP444LP	1-135
COP421L*	COP321L		NMOS Low Power	1.0k	64	19	Yes	No	3 Level	Yes	No	45 mW	9.9 mW	24	COP404LSN-5	COP444LP	1-135
COP422L*	COP322L		NMOS Low Power	1.0k	64	16	Yes	No	3 Level	Yes	No	45 mW	9.9 mW	20	COP404LSN-5	COP444LP	1-135
COP440	COP340		NMOS Hi Speed	2.0k	160	35	Yes	4 Sources	4 Level	Yes	Yes	205 mW	9.9 mW	40	COP404N	COP440R	1-181
COP441	COP341		NMOS Hi Speed	2.0k	160	23	Yes	4 Sources	4 Level	Yes	Yes	205 mW	9.9 mW	28	COP404N	COP440R	1-181
COP442	COP342		NMOS Hi Speed	2.0k	160	19	Yes	2 Sources	2 Level	Yes	No	205 mW	9.9 mW	24	COP404N	COP440R	1-181
COP444C*	COP344C	COP244C (Note 2)	CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	28	COP404CN	COP444CP	1-161
COP445C*	COP345C	COP245C (Note 2)	CMOS Hi Speed	2.0k	128	19	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	24	COP404CN	COP444CP	1-161
COP444L	COP344L		NMOS Low Power	2.0k	128	23	Yes	1 Source	3 Level	Yes	No	65 mW	9.9 mW	28	COP404LSN-6	COP444LP	1-204
COP445L	COP345L		NMOS Low Power	2.0k	128	19	Yes	No	3 Level	Yes	No	65 mW	9.9 mW	24	COP404LSN-6	COP444LP	1-204

Note 1: Datasheet found on page 1-8.

Note 2: Datasheet found on page 1-20.

*Microcontrollers available with Quick-Turns Prototype Post-Metal Programming (PMP).

COPS Family Development Tools

Commercial Temp Version 0°C to +70°C			Technology	Description		Features									Supplementary Description	Data Sheet Page
				Memory		I/O		Interrupt	Stack	Time Base Counter	Micro Bus	Typ. 5V Operat. Power	Max Standby at 3.3V	Size (Pins)		
				ROM (Bytes)	RAM (Digits)	I/O Pins	Serial I/O									
ROMless																
COP401L-X13			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	Has XTAL Oscillator Option	1-241
COP401L-R13			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	Has RC Oscillator Option	1-241
COP401L			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	ROMless Version of COP410L	1-227
COP402			NMOS Hi Speed	1.0k	63	20	Yes	1 Source	3 Level	Yes	No	50 mW	N/A mW	40	Has Interrupt, No Microbus	1-254
COP402M			NMOS Hi Speed	1.0k	63	16	Yes	Yes	3 Level	Yes	Yes	125 mW	N/A mW	40	No Interrupt, Has Microbus	1-254
COP404LSN-5			NMOS Low Power	1.0k	128	20	Yes	1 Source	3 Level	Yes	No	125 mW	N/A mW	40	W/Push-Pull Mem Interface	1-296
COP404			NMOS Hi Speed	2.0k	160	23	Yes	4 Sources	4 Level	Yes	Yes	35 mW	15 mW	48	ROMless Version of COP440	1-272
COP404C			CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	48	CMOS ROMless Device	1-279
PIGGYBACK																
COP420P			NMOS Hi Speed	1.0k	64	23	Yes	3 Sources	3 Level	Yes	No	50 mW	N/A mW	28	Includes: CPU, RAM, I/O	1-310
COP444LP			NMOS Low Power	2.0k	128	23	Yes	3 Sources	3 Level	Yes	No	125 mW	N/A mW	28	and EPROM Socket	1-310
COP444CP			CMOS Hi Speed	2.0k	128	23	Yes	1 Source	1 Level	Yes	Yes	1 mW	1 mW	28	Will Accept Standard EPROM	1-310

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On-Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains
DIAL-A-HELPER Users Manual P/N
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

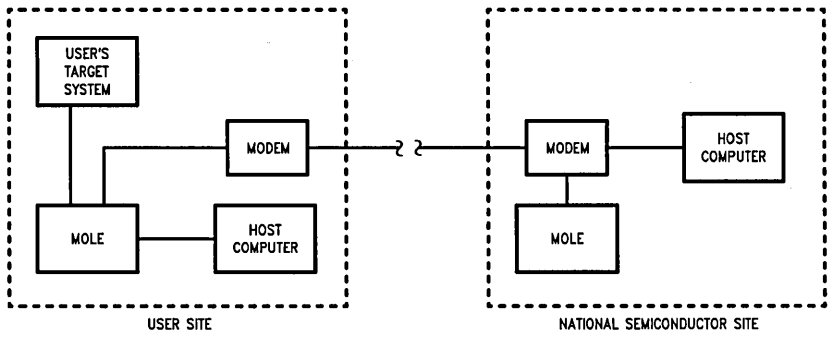
Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customers problem by actually getting customers system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Setup: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs. 7 Days

DIAL-A-HELPER



TL/XX/0072–1

National Semiconductor

COP210C/COP211C Single-Chip CMOS Microcontrollers

General Description

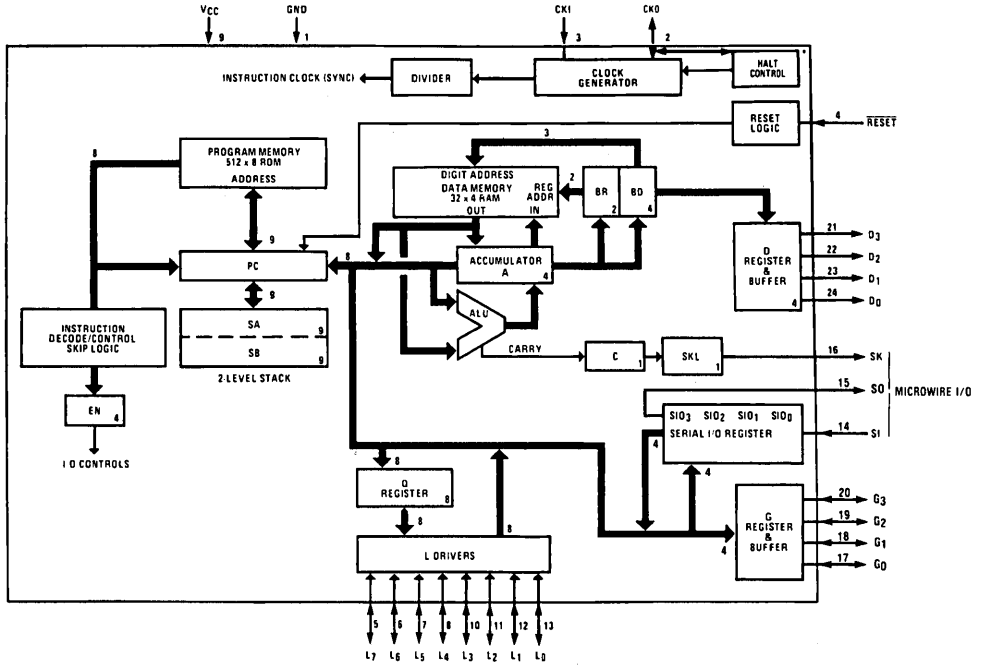
The COP210C and COP211C fully static, single-chip CMOS microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP211C is identical to the COP210C but with 16 I/O lines instead of 20. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller-oriented processor at a low end-product cost.

The COP404C should be used for exact emulation.

Features

- Lowest power dissipation (500 μW typical)
- Low cost
- Power-saving HALT mode with Continue function
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 20 I/O lines (COP210C)
- Two-level subroutine stack
- DC to 4.4 μs instruction time
- Single supply operation (4.5V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of the COP400 family
- Military temperature (−55°C to +125°C) devices

Block Diagram



TL/DD/8444-1

FIGURE 1. COP210C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Maximum Allowable Voltage	V_{CC} = 6V
Voltage at Any Pin	−0.3V to V_{CC} +0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Maximum Allowable Power Consumption	150 mW
Operating Temperature Range	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics −55°C ≤ T_A ≤ +125°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Supply Current (Note 1)	V_{CC} = 5.0V, t_c = Min (t_c is instruction cycle time)		4	mA
Power Supply Ripple (Notes 3, 4)	Peak to Peak		0.25	V
HALT Mode Current (Note 2)	V_{CC} = 5.0V, F_{IN} = 0 kHz		120	μA
Input Voltage Levels				
$\overline{\text{RESET}}$, CKI				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		−10	+10	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels	Standard Outputs (except CKO)			
LSTTL Operation	V_{CC} = 5.0V ±10%			
Logic High	I_{OH} = −100 μA	2.7		V
Logic Low	I_{OL} = 400 μA		0.6	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC}−0.2		V
Logic Low	I_{OL} = 10 μA		0.2	V
Allowable Sink/Source Current per Pin (Note 5)			5	mA
CKO Current Levels (As Clock Out)				
Sink ÷4	CKI = V_{CC}, V_{OUT} = V_{CC}	0.2		mA
÷8		0.4		mA
÷16		0.8		mA
Source ÷4	CKI = 0V, V_{OUT} = 0V	−0.2		mA
÷8		−0.4		mA
÷16		−0.8		mA
Allowable Loading on CKO (as HALT I/O pin)			50	pF
Current Needed to Override HALT (Note 6)				
To Continue	V_{IN} = 0.2 V_{CC}		2.0	mA
To Halt	V_{IN} = 0.7 V_{CC}		3.0	mA
TRI-STATE or Open Drain Leakage Current		−10	+10	μA

Note 1: Supply Current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}. L lines in TRI-STATE mode and tied to ground, all other outputs low and tied to ground.

Note 3: Voltage change must be less than 0.25V in a 1 ms period.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: SO Output sink current must be limited to keep V_{OL} less than 0.2 V_{CC}.

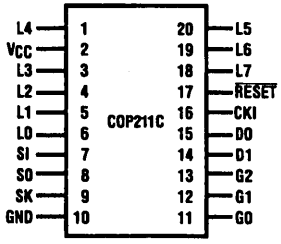
Note 6: When forcing HALT, current is only needed for a short time (approximatey 200 ns) to flip the HALT flip-flop.

AC Electrical Characteristics $-55°C \le T_A \le +125°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)		4.4	DC	μs
Operating CKI Frequency ÷4 mode		DC	0.9	MHz
÷8 mode		DC	1.8	MHz
÷16 mode		DC	3.6	MHz
Instruction Cycle Time RC Oscillator (Note 4)	R = 30k ±5%, C = 82 pF ±5% (÷4 Mode)	6	18	μs
Inputs (See *Figure 3*)				
t_{SETUP} (Note 4) G Inputs	$V_{CC} \ge 4.5V$	tc/4 + 0.8		μs
t_{SETUP} (Note 4) SI Input	$V_{CC} \ge 4.5V$	0.33		μs
t_{SETUP} (Note 4) All Others	$V_{CC} \ge 4.5V$	1.9		μs
t_{HOLD}		0.40		μs
Output Propagation Delay t_{PD1}, t_{PD0}	V_{OUT} = 1.5V, C_L = 100 pF, R_L = 5k		1.4	μs

Connection Diagrams

S.O. Wide and DIP



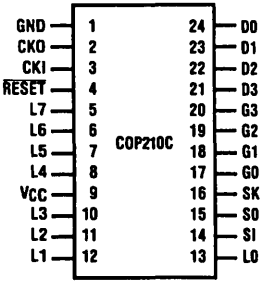
TL/DD/8444-2

Order Number COP211C-XXX/D,
See NS Hermetic Package Number D20A

Order Number COP211C-XXX/N,
See NS Molded Package Number N20A

Order Number COP211C-XXX/WM
See NS Surface Mount Package Number M20B

S.O. Wide and DIP



TL/DD/8444-3

Order Number COP210C-XXX/D,
See NS Hermetic Package Number D24C

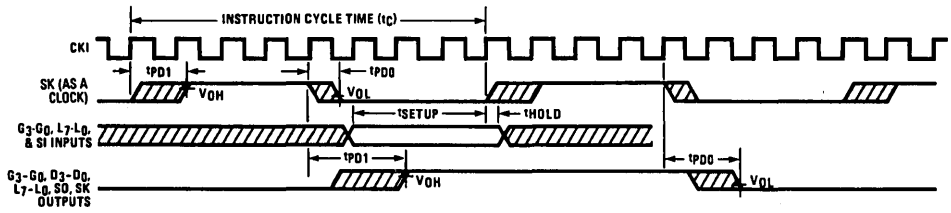
Order Number COP210C-XXX/N,
See NS Molded Package Number N24A

Order Number COP210C-XXX/WM
See NS Surface Mount Package Number M24B

Pin Descriptions

Pin	Description	Pin	Description
L_7–L_0	8-bit bidirectional I/O port with TRI-STATE	SK	Logic-controlled clock (or general purpose output)
G_3–G_0	4-bit bidirectional I/O port (G_2–G_0 for 20-pin package)	CKI	System oscillator input
D_3–D_0	4-bit general purpose output port (D_1–D_0 for 20-pin package)	CKO	Crystal oscillator output, or HALT mode I/O port (24-pin package only)
SI	Serial input (or counter input)	RESET	System reset input
SO	Serial output (or general purpose output)	V_{CC}	System power supply
		GND	System Ground

FIGURE 2



TL/DD/8444-4

FIGURE 3. Input/Output Timing Diagrams (Divide-by-8 Mode)

Functional Description

A block diagram of the COP210C is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP210C/211C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8 x 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

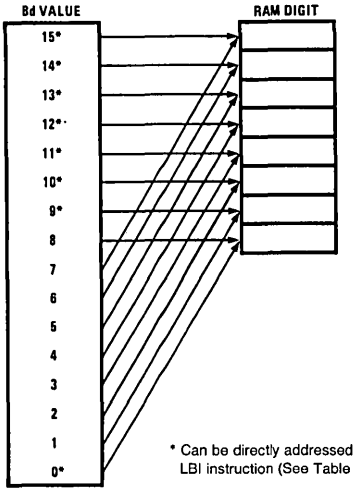
The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4.* The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP210C/211C is designed to ensure fully static operation of the device.

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP210C/211C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)



TL/DD/8444–5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP210C/211C is MICROWIRE compatible.

The D register provides four general purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.

Functional Description (Continued)

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

2. EN1 is not used, it has *no* effect on the COP210C/211C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

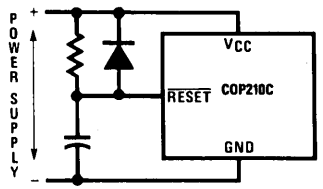
INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

When V_{CC} power is applied, the internal reset logic will keep the chip in initialization mode for up to 2500 instruction cycles. If the CKI clock is running at a low frequency, this could take a long time, therefore, the internal logic should be disabled by a mask option with initialization controlled solely by $\overline{\text{RESET}}$ pin.

Note: If CKI clock is less than 32 kHz, the internal reset logic (Option 25 = 1) *must* be disabled and the external RC network *must* be present.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/8444-6

RC > 5 × Power Supply Rise Time and RC > 100 × CKI Period

FIGURE 5. Power-Up Clear Circuit

COP211C

If the COP210C is bonded as a 20-pin package, it becomes the COP211C, illustrated in *Figure 2,* COP210C/211C Connection Diagrams. Note that the COP211C does not contain D2, D3, G3, or CKO. Use of this option, of course, precludes use of D2, D3, G3, and CKO options. All other options are available for the COP211C.

HALT MODE

The COP210C/211C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip also may be halted by the HALT instruction or by forcing CKO high when it is used as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode has slight differences depending upon the type of oscillator used.

a. 1-pin oscillator—RC or external

The HALT mode may be entered into by either program control (HALT instruction) or by forcing CKO to a logic "1" state.

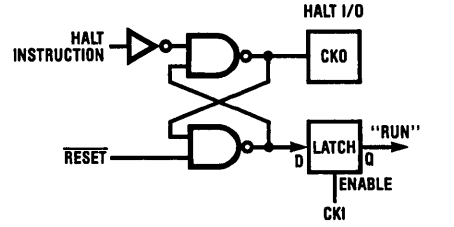
The circuit may be awakened by one of two different methods:

1) Continue function. By forcing CKO to a logic "0", the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.

2) Restart. Forcing the $\overline{\text{RESET}}$ pin to a logic "0" will restart the chip regardless of HALT or CKO (see initialization).

b. 2-pin oscillator—crystal

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.



TL/DD/8444-7

Halt I/O Port

CKO PIN OPTIONS

In a crystal-controlled oscillator system, CKO is used as an output to the crystal network. CKO will be forced high during the execution of a HALT instruction, thus inhibiting the crystal network. If a 1-pin oscillator system is chosen (RC or

Functional Description (Continued)

external), CKO will be selected as HALT and is an I/O flip-flop which is an indicator of the HALT status. An external signal can override this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and the CKO output will go high to keep the chip stopped. By forcing a low level to CKO, the chip will continue and CKO output will go low.

All features associated with the CKO I/O pin are available with the 24-pin package only.

OSCILLATOR OPTIONS

There are three options available that define the use of CKI and CKO.

a. Crystal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optionally by 8 or 4).

b. External Oscillator. CKI is configured as LSTTL-compatible input accepting an external clock signal. The external frequency is divided by 16 (optionally by 8 or 4) to give the instruction cycle time. CKO is the HALT I/O port.

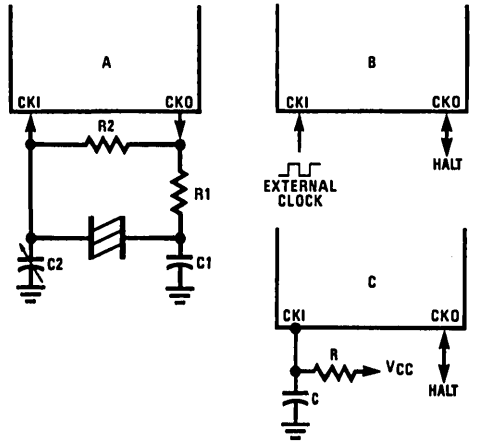
c. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.

COP210C/COP211C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP210C/211C instruction set.



TL/DD/8444-8

FIGURE 6. COP210C Oscillator

Crystal or Resonator					RC-Controller Oscillator		
Crystal Value	Component Values R1	R2	C1pF	C2pF	R	C	Cycle Time
32 kHz	220k	20M	30	5-36	47k	100 pF	17-25 μs
455 kHz	5k	10M	80	40	30k	82 pF	6-18 μs
3.58 MHz	1k	1M	30	6-36	Note: 15k≤R≤150k, 50 pF≤C≤150 pF		

TABLE II. COP210C/211C Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

Instruction Set (Continued)

TABLE III. COP210C/211C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– –	|0110|000|a_8| |$a_{7:0}$|	a → PC	None	Jump
JP	a	–	|1| $a_{6:0}$ | (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 1)
		–	|11| $a_{5:0}$ | (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	|10| $a_{5:0}$ |	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 2)
JSR	a	6– –	|0110|100|a_8| |$a_{7:0}$|	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	|0011|0011| |0011|1000|		None	Halt processor

Instruction Set (Continued)

TABLE III. COP210C/211C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	|0011|0011| |0011|1100|	$A \rightarrow Q_{7:4}$ $RAM(B) \rightarrow Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	|0011|0011| |0010|1100|	$Q_{7:4} \rightarrow RAM(B)$ $Q_{3:0} \rightarrow A$	None	Copy Q to RAM, A
LD	r	−5	|00|r|0101|	$RAM(B) \rightarrow A$ $Br \oplus r \rightarrow Br$	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	|1011|1111|	$ROM(PC_8,A,M) \rightarrow Q$ $SA \rightarrow SB$	None	Load Q Indirect
RMB	0 1 2 3	4C 45 42 43	|0100|1100| |0100|0101| |0100|0010| |0100|0011|	$0 \rightarrow RAM(B)_0$ $0 \rightarrow RAM(B)_1$ $0 \rightarrow RAM(B)_2$ $0 \rightarrow RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	|0100|1101| |0100|0111| |0100|0110| |0100|1011|	$1 \rightarrow RAM(B)_0$ $1 \rightarrow RAM(B)_1$ $1 \rightarrow RAM(B)_2$ $1 \rightarrow RAM(B)_3$	None	Set RAM Bit
STII	y	7−	|0111| y |	$y \rightarrow RAM(B)$ $Bd + 1 \rightarrow Bd$	None	Store Memory Immediate and Increment Bd
X	r	−6	|00|r|0110|	$RAM(B) \longleftrightarrow A$ $Br \oplus r \rightarrow Br$	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	|0010|0011| |1011|1111|	$RAM(3,15) \longleftrightarrow A$	None	Exchange A with RAM (3,15)
XDS	r	−7	|00|r|0111|	$RAM(B) \longleftrightarrow A$ $Bd - 1 \rightarrow Bd$ $Br \oplus r \rightarrow Br$	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	−4	|00|r|0100|	$RAM(B) \longleftrightarrow A$ $Bd + 1 \rightarrow Bd$ $Br \oplus r \rightarrow Br$	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	$A \rightarrow Bd$	None	Copy A to Bd
CBA		4E	|0100|1110|	$Bd \rightarrow A$	None	Copy Bd to A
LBI	r,d	−	|00|r|(d - 1)| (d = 0,9:15)	$r,d \rightarrow B$	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6−	|0011|0011| |0110| y |	$y \rightarrow EN$	None	Load EN Immediate

Instruction Set (Continued)

TABLE III. COP210C/211C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 | 0011		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
		21	0010 | 0001			
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0 = 0$	
	1	11	0001 | 0001	2nd byte	$G_1 = 0$	
	2	03	0000 | 0011	2nd byte	$G_2 = 0$	
	3	13	0010 | 0011	2nd byte	$G_3 = 0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1 = 0$	
	2	03	0000 | 0011		$RAM(B)_2 = 0$	
	3	13	0001 | 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 | 0011	G → A	None	Input G Ports to A
		2A	0010 | 1010			
INL		33	0011 | 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 | 1110	$L_{3:0}$ → A		
OBD		33	0011 | 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 | 1110			
OMG		33	0011 | 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP210C/211C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register). If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

Description of Selected Instructions (Continued)

INSTRUCTION SET NOTES

a. The first word of a COP210C/211C program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).

c. The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw additional current. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP210C current drain.

$Ic = Iq + (V \times 35 \times Fi) + (V \times 2195 \times Fi/Dv)$

where Ic = chip current drain in microamps

Iq = quiescent leakage current (from curve)

Fi = CKI frequency in megahertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 4),

$Ic = 10 + (5 \times 35 \times 0.4) + (5 \times 2195 \times 0.4/4)$

$Ic = 10 + 50 + 1097.5 = 1157.5\ \mu A$

I/O OPTIONS

COP210C/211C outputs have the following optional configurations, illustrated in *Figure 7*:

a. Standard. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

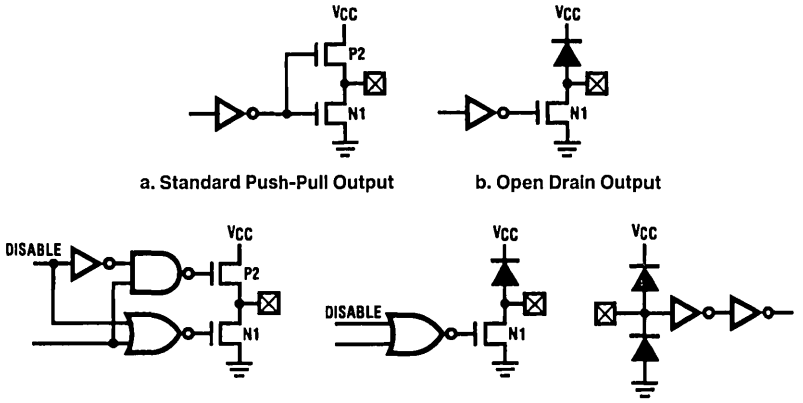
b. Open Drain. An N-channel device to ground only, allowing external pull-up as required by the user's application.

c. Standard TRI-STATE L Output. A CMOS output buffer similar to (a) which may be disabled by program control.

d. Open-Drain TRI-STATE L Output. This has the N-channel device to ground only.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (*Figure 7e*).

When using either the G or L I/O ports as inputs, an external pull-up device is necessary.



a. Standard Push-Pull Output

b. Open Drain Output

c. Standard TRI-STATE "L" Output

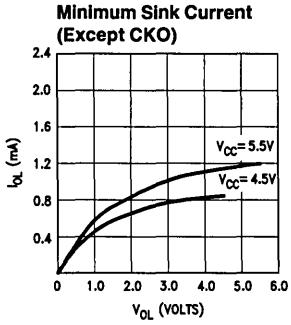
d. Open Drain TRI-STATE "L" Output

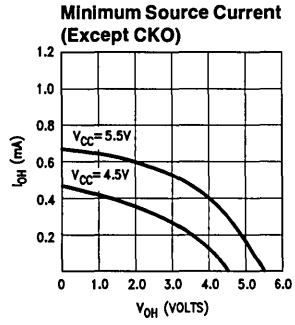
e. Hi-Z Input

FIGURE 7. I/O Configurations

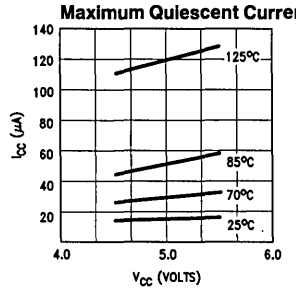
All output drivers uses one or two common devices numbered 1 to 2. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.

Typical Performance Characteristics





TL/DD/8444–10



TL/DD/8444–11

FIGURE 8

Option List

The COP210C/211C mask-programmable options are assigned numbers which correspond with the COP210C pins.

The following is a list of COP210C options. When specifying a COP211 chip, options 20, 21, and 22 must be set to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: 0 = Ground Pin. No options available.

Option 2: CKO I/O Port Determined by Option 3. = 0 no option (a. is crystal oscillator output for two pin oscillator b. is HALT I/O for one pin oscillator)

Option 3: CKI Input.
- = 0: Crystal-controlled oscillator input (÷ 4).
- = 1: Single-pin RC-controlled oscillator (÷ 4).
- = 2: External oscillator input (÷ 4).
- = 3: Crystal oscillator input (÷ 8).
- = 4: External oscillator input (÷ 8).
- = 5: Crystal oscillator input (÷ 16).
- = 6: External oscillator input (÷ 16).

Option 4: $\overline{\text{RESET}}$ Input = 1: Hi-Z input. No option available.

Option 5: L_7 Driver
- = 0: Standard TRI-STATE push-pull output.
- = 2: Open-drain TRI-STATE output.

Option 6: L_6 Driver. (Same as Option 5.)

Option 7: L_5 Driver. (Same as Option 5.)

Option 8: L_4 Driver. (Same as Option 5.)

Option 9: V_{CC} Pin = 0 no option.

Option 10: L_3 Driver. (Same as Option 5.)

Option 11: L_2 Driver. (Same as Option 5.)

Option 12: L_1 Driver. (Same as Option 5.)

Option 13: L_0 Driver. (Same as Option 5.)

Option 14: SI Input.
- No option available.
- = 1: Hi-Z input.

Option 15: SO Output.
- = 0: Standard push-pull output.
- = 2: Open-drain output.

Option 16: SK Driver. (Same as Option 15.)

Option 17: G_0 I/O Port. (Same as Option 15.)

Option 18: G_1 I/O Port. (Same as Option 15.)

Option 19: G_2 I/O Port. (Same as Option 15.)

Option 20: G_3 I/O Port. (Same as Option 15.)

Option 21: D_3 Output. (Same as Option 15.)

Option 22: D_2 Output. (Same as Option 15.)

Option 23: D_1 Output. (Same as Option 15.)

Option 24: D_0 Output. (Same as Option 15.)

Option 25: Internal Initialization Logic.
- = 0: Normal operation.
- = 1: No internal initialization logic.

Option 26: No option available.

Option 27: COP Bonding
- = 0: COP210C (24-pin device).
- = 1: COP211C (20-pin device). See Note.
- = 2: COP210C and COP211C. See Note.

Note: If option 27 = 1 or 2 then option 20 must = 0.

Option Table

Please fill out a photocopy of the Option Table and send along with your EPROM.

Option Table

Option 1 Value =	0	is: Ground Pin	
Option 2 Value =	0	is: CKO Pin	
Option 3 Value =		is: CKI Input	
Option 4 Value =	1	is: $\overline{\text{RESET}}$ Input	
Option 5 Value =		is: L_7 Driver	
Option 6 Value =		is: L_6 Driver	
Option 7 Value =		is: L_5 Driver	
Option 8 Value =		is: L_4 Driver	
Option 9 Value =	0	is: V_{CC} Pin	
Option 10 Value =		is: L_3 Driver	
Option 11 Value =		is: L_2 Driver	
Option 12 Value =		is: L_1 Driver	
Option 13 Value =		is: L_0 Driver	
Option 14 Value =	1	is: SI Input	
Option 15 Value =		is: SO Output	
Option 16 Value =		is: SK Driver	
Option 17 Value =		is: G_0 I/O Port	
Option 18 Value =		is: G_1 I/O Port	
Option 19 Value =		is: G_2 I/O Port	
Option 20 Value =		is: G_3 I/O Port	
Option 21 Value =		is: D_3 Output	
Option 22 Value =		is: D_2 Output	
Option 23 Value =		is: D_1 Output	
Option 24 Value =		is: D_0 Output	
Option 25 Value =		is: Internal Initialization Logic	
Option 26 Value =	0	is: No Option	
Option 27 Value =		is: COPS Bonding	

National Semiconductor

COP224C/COP225C/COP226C/COP244C/COP245C Single-Chip 1k and 2k CMOS Microcontrollers

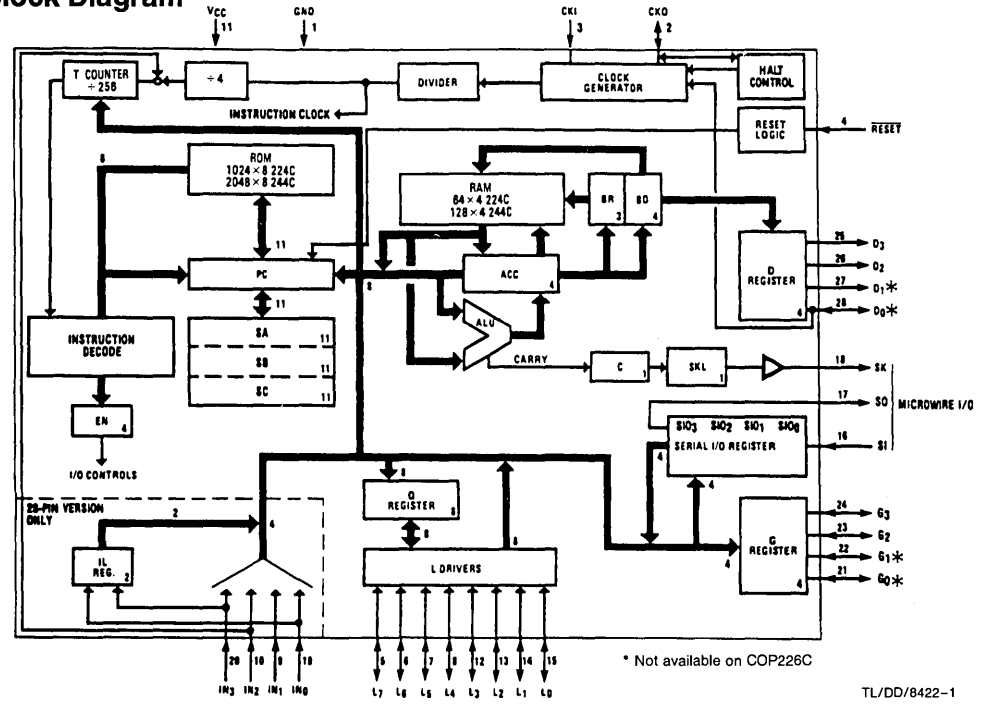
General Description

The COP224C, COP225C, COP226C, COP244C and COP245C fully static, Single-Chip CMOS Microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon gate microCMOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP224C and COP244C are 28 pin chips. The COP225C and COP245C are 24-pin versions (4 inputs removed) and COP226C is 20-pin version with 15 I/O lines. Standard test procedures and reliable high-density techniques provide the medium to large volume customers with a customized microcontroller at a low end-product cost. These microcontrollers are appropriate choices in many demanding control environments especially those with human interface.

Features

- Lowest power dissipation (600 μW typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4.4 μs instruction time
- 2k x 8 ROM, 128 x 4 RAM (COP244C/COP245C)
- 1k x 8 ROM, 64 x 4 RAM (COP224C/COP225C/COP226C)
- 23 I/O lines (COP244C and COP224C)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (4.5V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS output compatible
- Software/hardware compatible with COP400 family
- Military temperature (−55°C to +125°C) operation

Block Diagram



* Not available on COP226C

TL/DD/8422–1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at any Pin	−0.3V to V_{CC} +0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Total Allowable Power Dissipation	150 mW
Operating Temperature Range	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-55°C \le T_A \le +125°C$, $+4.5V \le V_{CC} \le +5.5V$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Power Supply Ripple (Note 5)	Peak to Peak		0.25 V_{CC}	V
Supply Current (Note 1)	V_{CC} = 5.0V, tc = 4.4 μs (tc is instruction cycle time)		5	mA
HALT Mode Current (Note 2)	V_{CC} = 5.0V, F_{IN} = 0 kHz		200	μA
Input Voltage Levels				
$\overline{RESET}$, CKI, D_0 (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		−10	+10	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels (except CKO)	Standard Outputs			
LSTTL Operation	V_{CC} = 5.0V ± 10%			
Logic High	I_{OH} = −100 μA	2.7		V
Logic Low	I_{OL} = 400 μA		0.6	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC} −0.2		V
Logic Low	I_{OL} = 10 μA		0.2	V
CKO Current Levels (As Clock Out)				
Sink ÷4	CKI = V_{CC}, V_{OUT} = V_{CC}	0.2		mA
Sink ÷8		0.4		mA
Sink ÷16		0.8		mA
Source ÷4	CKI = 0V, V_{OUT} = 0V	−0.2		mA
Source ÷8		−0.4		mA
Source ÷16		−0.8		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			50	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	V_{IN} = 0.2 V_{CC}		2.0	mA
To Halt	V_{IN} = 0.7 V_{CC}		3.0	mA
TRI-STATE or Open Drain Leakage Current		−10	+10	μA

AC Electrical Characteristics $-55°C \leq T_A \leq +125°C$, $+4.5V \leq V_{CC} \leq +5.5V$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (tc)		4.4	DC	μs
Operating CKI ÷4 mode Frequency ÷8 mode ÷16 mode		DC DC DC	0.9 1.8 3.6	MHz MHz MHz
Duty Cycle (Note 4)	$f_1 = 3.6$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 3.6$ MHz External Clock		60	ns
Fall Time (Note 4)	$f_1 = 3.6$ MHz External Clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)	R = 30k ±5% C = 82 pF ±5% (÷4 Mode)	6	18	μs
Inputs: (See *Figure 3*) (Note 4)				
t_{SETUP}	G Inputs	tc/4 + 0.8		μs
	SI Input	0.33		μs
	All Others	1.9		μs
t_{HOLD}		0.4		μs
Output Propagation Delay t_{PD1}, t_{PD0}	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5k$		1.4	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}, L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

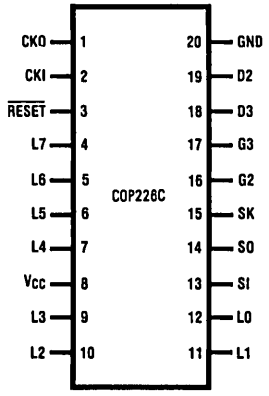
Note 4: This parameter is not tested but guaranteed by design. Variation due to the device included.

Note 5: Voltage change must be less than 0.25 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Connection Diagrams

S.O. Wide and DIP



Top View

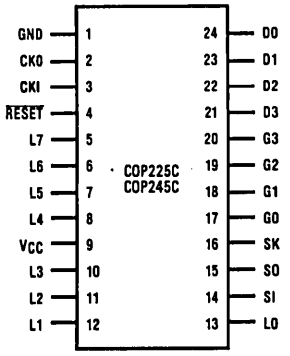
TL/DD/8422-2

Order Number COP226C-XXX/N
See NS Molded Package Number N20A

Order Number COP226C-XXX/D
See NS Hermetic Package Number D20A

Order Number COP226C-XXX/WM
See NS Surface Mount Package Number M20B

S.O. Wide and DIP



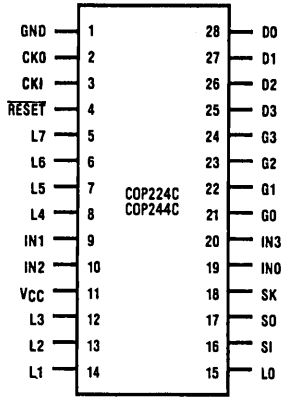
Top View

TL/DD/8422-3

Order Number COP225C-XXX/N
or COP245C-XXX/N
See NS Molded Package Number N24A

Order Number COP225C-XXX/D
or COP245C-XXX/D
See NS Hermetic Package Number D24C

DIP



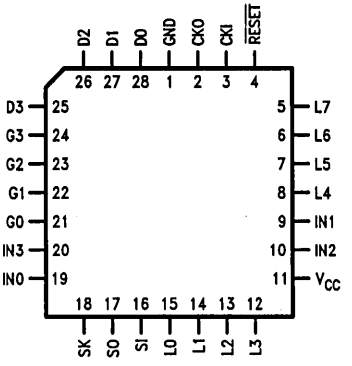
Top View

TL/DD/8422-4

Order Number COP224C-XXX/N
or COP244C-XXX/N
See NS Molded Package Number N28B

Order Number COP224C-XXX/D
or COP244C-XXX/D
See NS Hermetic Package Number D28C

PLCC



TL/DD/8422-13

Order Number COP224C-XXX/V
or COP244C-XXX/V
See NS PLCC Package Number V28A

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L7–L0	8-bit bidirectional port with TRI-STATE	SK	Logic controlled clock output
G3–G0	4-bit bidirectional I/O port	CKI	Chip oscillator input
D3–D0	4-bit output port	CKO	Oscillator output, HALT I/O port or general purpose input
IN3–IN0	4-bit input port (28 pin package only)	$\overline{\text{RESET}}$	Reset input
SI	Serial input or counter input	V_{CC}	Most positive power supply
SO	Serial or general purpose output	GND	Ground

Functional Description

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

Caution:

The output options available on the COP224C/225C/226C and COP244C/245C are not the same as those available on the COP324C/325C/326C, COP344C/345C, COP424C/425C/426C and COP444C/445C. Options not available on the COP224C/225C/226C and COP244C/245C are: Option 2 value 2; Option 4 value 0; Option 5 value 1; Option 9 value 0; Option 17 value 1; Option 30, Dual Clock, all values; Option 32, Microbus™, all values; Option 33 values 2, 4, and 6; Option 34 all values; and Option 35 all values.

PROGRAM MEMORY

Program Memory consists of ROM, 1024 bytes for the COP224C/225C/226C and 2048 bytes for the COP244C/245C. These bytes of ROM may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by an 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

Data memory consists of a 512-bit RAM for the COP244C/245C, organized as 8 data registers of 16 × 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register.

Data memory consists of a 256-bit RAM for the COP224C/225C/226C, organized as 4 data registers of 16 × 4-bits digits. The B register is 6 bits long. Upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions.

The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, to input 4 bits of L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes depending on a mask-programmable option: as a timer or as an external event counter. When the T counter overflows, an

Functional Description (Continued)

overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 7*.

Four general-purpose inputs, IN3–IN0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

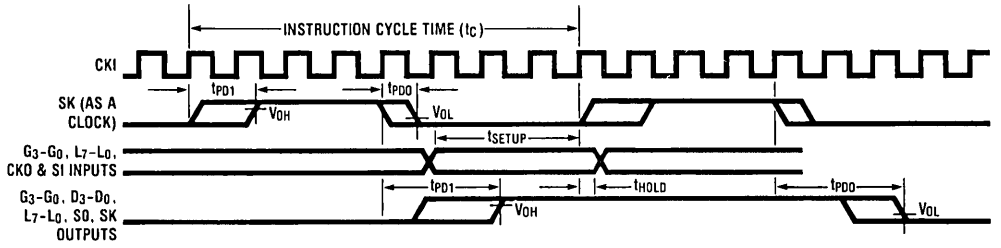
0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.
3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.

b. An interrupt will be recognized only on the following conditions:
   1. EN1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN_1 input.
   3. A currently executing instruction has been completed.



TL/DD/8422–5

FIGURE 3. Input/Output Timing Diagrams (divide by 8 mode)

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1,SK=clock If SKL=0,SK=0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL=1,SK=clock If SKL=0,SK=0
1	0	Binary Counter	Input to Counter	0	SK=SKL
1	1	Binary Counter	Input to Counter	1	SK=SKL

Functional Description (Continued)

4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

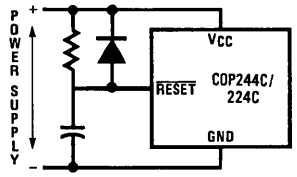
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 4* must be connected to the $\overline{\text{RESET}}$ pin (the conditions in *Figure 4* must be met). The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

Note: If CKI clock is less than 32 kHz, the internal reset logic (option #29 = 1) MUST be disabled and the external RC circuit must be used.



TL/DD/8422-6

FIGURE 4. Power-Up Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).

TIMER

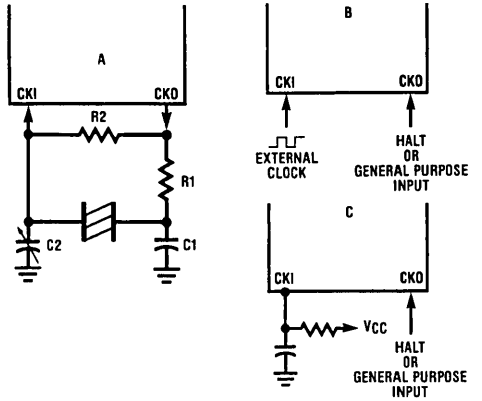
There are two modes selected by mask option:

a. Time-base counter. In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset.

For example, using a 3.58 MHz crystal with a divide-by-16 option, the instruction cycle frequency of 223.70 kHz increments the 10-bit timer every 4.47 μs. By presetting the counter and detecting overflow, accurate timeouts between 17.88 μs (4 counts) and 4.577 ms (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

b. External event counter. In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

Note: The IT instruction is not allowed in this mode.



TL/DD/8422-7

Crystal or Resonator

Crystal Value	Component Values			
	R1	R2	C1(pF)	C2(pF)
32 kHz	220k	20M	30	6–36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6–36
3.6 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R	C	Cycle Time	V_{CC}
30k	82 pF	6–18 μs	≥4.5V

Note: 15k≤R≤150k
50 pF≤C≤150 pF

FIGURE 5. Oscillator Component Values

Functional Description (Continued)

HALT MODE

The COP244C/245C/224C/225C/226C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by the HALT instruction or by forcing CKO high when it is mask-programmed as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The chip may be awakened by one of two different methods:

- Continue function: by forcing CKO low, if it mask-programmed as a HALT I/O port, the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization).

The HALT mode is the minimum power dissipation state.

CKO PIN OPTIONS

a. Two-pin oscillator—(Crystal). See *Figure 6a.*

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. The HALT mode may be entered by program control (HALT instruction) which forces CKO high, thus inhibiting the crystal network. The circuit can be awakened only by forcing the $\overline{\text{RESET}}$ pin to a logic "0" (restart).

b. One-pin oscillator—(RC or external). See *Figure 6b.*

If a one-pin oscillator system is chosen, two options are available for CKO:

- CKO can be selected as the HALT I/O port. In that case, it is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and CKO output will stay high to keep the chip stopped if the external driver returns to high impedance state. By forcing a low level to CKO, the chip will continue and CKO will stay low.
- As another option, CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction.

OSCILLATOR OPTIONS

There are three basic clock oscillator configurations available as shown by *Figure 5.*

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency optionally divided by 4, 8 or 16.

b. External Oscillator. The external frequency is optionally divided by 4, 8 or 16 to give the instruction cycle time. CKO is the HALT I/O port or a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port or a general purpose input.

Figure 7 shows the clock and timer diagram.

COP245C AND COP225C 24-PIN PACKAGE OPTION

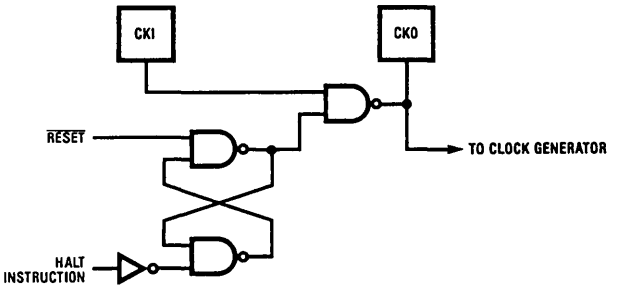
If the COP244C/224C is bonded in a 24-pin package, it becomes the COP245C/225C, illustrated in *Figure 2*, Connection diagrams. Note that the COP245C/225C does not contain the four general purpose IN inputs (IN3–IN0). Use of this option precludes, of course, use of the IN options, interrupt feature, external event counter feature.

Note: If user selects the 24-pin package, options 9, 10, 19 and 20 must be selected as a "2". See option list.

COP226C 20-PIN PACKAGE OPTION

If the COP225C is bonded as 20-pin device it becomes the COP226C. Note that the COP226C contains all the COP225C pins except D_0, D_1, G_0, and G_1.

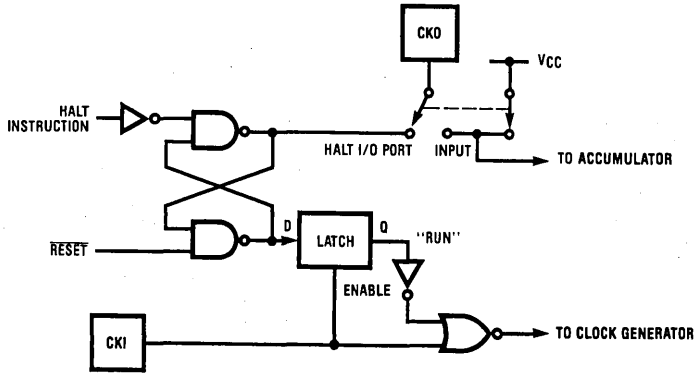
Block Diagram



TL/DD/8422–8

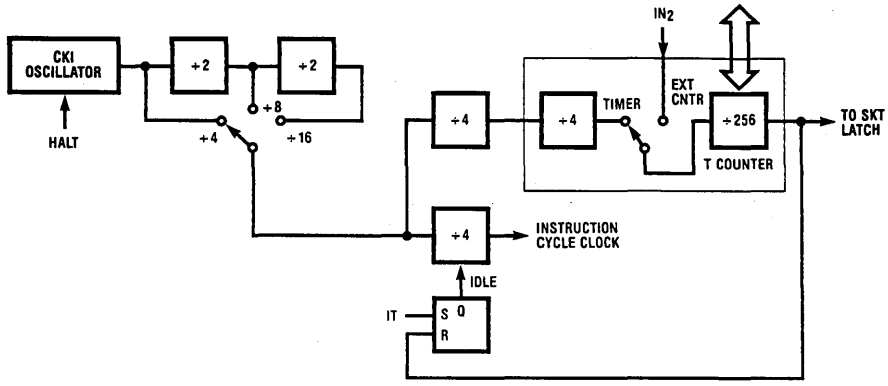
FIGURE 6a. Halt Mode—Two-Pin Oscillator

Block Diagrams (Continued)



TL/DD/8422-9

FIGURE 6b. Halt Mode—One-Pin Oscillator



TL/DD/8422-10

FIGURE 7. Clock and Timer

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

TABLE II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit accumulator
B	7-bit RAM address register (6-bit for COP224C)
Br	Upper 3 bits of B (register address) (2-bit for COP224C)
Bd	Lower 4 bits of B (digit address)
C	1-bit carry register
D	4-bit data output port
EN	4-bit enable register
G	4-bit general purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA,SB,SC	11-bit 3-level subroutine stack
SIO	4-bit shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Instruction Operand Symbols	
d	4-bit operand field, 0–15 binary (RAM digit select)
r	3(2)-bit operand field, 0–7(3) binary (RAM register select)
a	11-bit operand field, 0–2047 (1023)
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x
Operational Symbols	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	One's complement of A
⊕	Exclusive-or
:	Range of values

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

TABLE III. COP244C/245C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A+RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	$A+10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5−	|0101| y |	A+y → A	Carry	Add Immediate. Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$+RAM(B)+C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A} \rightarrow A$	None	Ones complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A⊕RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM ($PC_{10:8}$ A,M) → $PC_{7:0}$	None	Jump Indirect (Notes 1, 3)
JMP	a	6–	0110 | 0 | $a_{10:8}$	a → PC	None	Jump
		– –	$a_{7:0}$			
JP	a	– –	1 | $a_{6:0}$ (pages 2, 3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 4)
			or			
		– –	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 | $a_{5:0}$	PC+1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6–	0110 | 1 | $a_{10:8}$	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
		– –	$a_{7:0}$			
RET		48	0100 | 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33	0011 | 0011		None	HALT Processor
		38	0011 | 1000			
IT		33	0011 | 0011			IDLE till Timer
		39	0011 | 1001		None	Overflows then Continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33	0011 | 0011	A → $T_{7:4}$		
		3F	0011 | 1111	RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CTMA		33	0011 | 0011	$T_{7:4}$ → RAM(B)		
		2F	0010 | 1111	$T_{3:0}$ → A	None	Copy T to RAM, A
CAMQ		33	0011 | 0011	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 | 1100	RAM(B) → $Q_{3:0}$		
CQMA		33	0011 | 0011	$Q_{7:4}$ → RAM(B)	None	Copy Q to RAM, A
		2C	0010 | 1100	$Q_{3:0}$ → A		
LD	r	–5	00 | r | 0101 (r=0:3)	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23	0010 | 0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		– –	0 | r | d			
LQID		BF	1011 | 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7–	|0111| y |	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate 1 and Increment Bd
X	r	–6	|00| r |0110| (r=0:3)	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 – –	|0010|0011| |1| r | d |	RAM(r,d) ⟷ A	None	Exchange A with RAM Pointed to Directly by r,d
XDS	r	–7	|00|r|0111| (r=0:3)	RAM(B) ⟷ A Bd–1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	–4	|00| r |0100| (r=0:3)	RAM(B) ⟷ A Bd+1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	– –	|00|r|(d–1)| (r=0:3: d=0,9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33 – –	|0011|0011| |1| r | d | (any r, any d)			
LEI	y	33 6–	|0011|0011| |0110| y |	y → EN	None	Load EN Immediate (Note 7)
XABR		12	|0001|0010|	A ⟷ Br	None	Exchange A with Br (Note 8)
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C="1"	Skip if C is True
SKE		21	|0010|0001|		A=RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|		$G_0=0$	
	1	11	|0001|0001|		$G_1=0$	
	2	03	|0000|0011|	2nd byte	$G_2=0$	
	3	13	|0001|0011|		$G_3=0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1=0$	
	2	03	|0000|0011|		$RAM(B)_2=0$	
	3	13	|0001|0011|		$RAM(B)_3=0$	
SKT		41	|0100|0001|		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 | 0011	G → A	None	Input G Ports to A
		2A	0010 | 1010			
ININ		33	0011 | 0011	IN → A	None	Input IN Inputs to A (Note 2)
		28	0010 | 1000			
INIL		33	0011 | 0011	IL_3, CKO,"0", IL_0 → A	None	Input IL Latches to A (Note 3)
		29	0010 | 1001			
INL		33	0011 | 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM,A
		2E	0010 | 1110	$L_{3:0}$ → A		
OBD		33	0011 | 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 | 1110			
OGI	y	33	0011 | 0011	y → G	None	Output to G Ports Immediate
		5–	0101 | y			
OMG		33	0011 | 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin packages since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B(Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: For 2K ROM devices, A ⟷ Br (0 → A3). For 1K ROM devices, A ⟷ Br (0,0 → A3, A2).

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10:PC8,A,M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC+1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC7:4, RAM(B) → PC3:0, leaving PC10, PC9 and PC8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10:8,A,M. PC10,PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT   ; load T counter
SKT    ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is mask-programmed as an external event counter (option #31 = 1).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKO and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A0 is input into A1. IL latches are cleared on reset. IL latches are not available on the COP245C/225C, and COP226C.

INSTRUCTION SET NOTES

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.

c. The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Note: The COP224C/225C/226C needs only 10 bits to address its ROM. Therefore, the eleventh bit (P10) is ignored.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal or resonator generated clock input will draw additional current. For example, a 500 kHz crystal input will typically draw 100 μA more than a squarewave input. An R/C oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate operating current drain.

$I_{CO} = I_Q + V \times 70 \times Fi + V \times 2400 \times Fi/Dv$ where:

I_{CO} = chip operating current drain in microamps

I_Q = quiescent leakage current (from curve)

Fi = CKI frequency in MegaHertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example at 5 volts V_{CC} and 400 kHz (divide by 4)

$I_{CO} = 120 + 5 \times 70 \times 0.4 + 5 \times 2400 \times 0.4/4$

$I_{CO} = 120 + 140 + 1200 = 1460\ \mu A$

Power Dissipation (Continued)

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$Ici = I_Q + V \times 70 \times Fi$

For example, at 5 volts V_{CC} and 400 kHz

$Ici = 120 + 5 \times 70 \times 0.4 = 260\ \mu A$

The total average current will then be the weighted average of the operating current and the idle current:

$$Ita = I_{CO} \times \frac{To}{To+Ti} + Ici \times \frac{Ti}{To+Ti}$$

where: Ita = total average current
I_{CO} = operating current
Ici = idle current
To = operating time
Ti = idle time

I/O OPTIONS

Outputs have the following optional configurations, illustrated in *Figure 8*:

a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

b. Open Drain — An N-channel device to ground only, allowing external pull-up as required by the user's application.

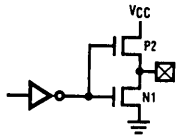
c. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control.

d. Open-Drain TRI-STATE L Output — This has the N-channel device to ground only.

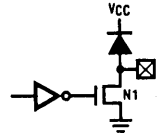
All inputs have the following option:

e. Hi-Z input which must be driven by the users logic.

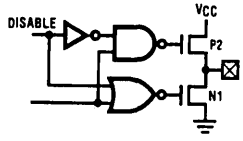
All output drivers use two common devices numbered 1 to 2. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 9* for each of these devices to allow the designer to effectively use these I/O configurations.



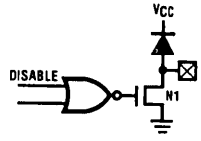
a. Standard Push-Pull Output



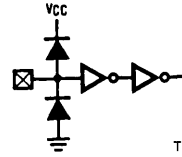
b. Open-Drain Output



c. Standard TRI-STATE "L" Output



d. Open Drain TRI-STATE "L" Output

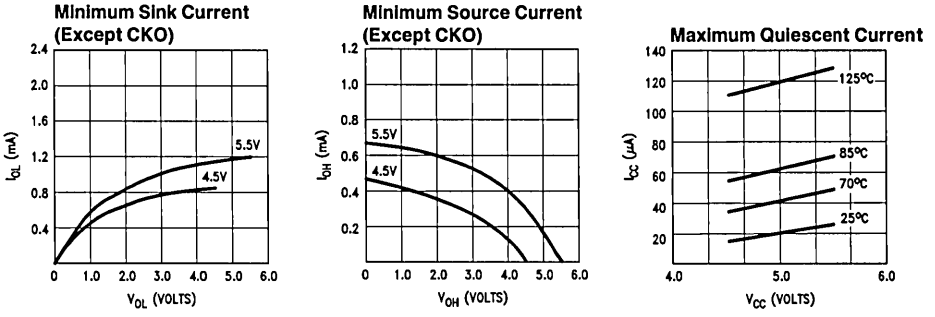


TL/DD/8422–11

e. Hi-Z Input

FIGURE 8. Input/Output Configurations

Power Dissipation (Continued)



TL/DD/8422–12

FIGURE 9. Input/Output Characteristics

Option List

The COP244C/245C/224C/225C/COP226C mask-programmable options are assigned numbers which correspond with the COP244C/224C pins.

The following is a list of options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Caution:
The output options available on the COP224C/225C/226C and COP244C/245C are not the same as those available on the COP324C/325C/326C, COP344C/345C, COP424C/425C/426C and COP444C/445C. Options not available on the COP224C/225C/226C and COP244C/245C are: Option 2 value 2; Option 4 value 0; Option 5 value 1; Option 9 value 0; Option 17 value 1; Option 30, Dual Clock, all values; Option 32, Microbus, all values; Option 33 values 2 4, and 6; Option 34 all values; and Option 35 all values.

PLEASE FILL OUT THE OPTION TABLE on the next page. Photocopy the option data and send it in with your disk or EPROM.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Pin
- = 0: clock generator output to crystal/resonator
- = 1: HALT I/O port
- = 3: general purpose input, high-Z

Option 3: CKI input
- = 0: Crystal controlled oscillator input divide by 4
- = 1: Crystal controlled oscillator input divide by 8
- = 2: Crystal controlled oscillator input divide by 16
- = 4: Single-pin RC controlled oscillator (divide by 4)
- = 5: External oscillator input divide by 4
- = 6: External oscillator input divide by 8
- = 7: External oscillator input divide by 16

Option 4: $\overline{\text{RESET}}$ input
- = 1: Hi-Z input

Option 5: L7 Driver
- = 0: Standard TRI-STATE push-pull output
- = 2: Open-drain TRI-STATE output

Option 6: L6 Driver — (same as option 5)

Option 7: L5 Driver — (same as option 5)

Option 8: L4 Driver — (same as option 5)

Option 9: IN1 input
- = 1: Hi-Z input, mandatory for 28 Pin Package
- = 2: Mandatory for 20 and 24 Pin Packages

Option 10: IN2 input — (same as option 9)

Option 11 = 0: V_{CC} Pin — no option available

Option 12: L3 Driver — (same as option 5)

Option 13: L2 Driver — (same as option 5)

Option 14: L1 Driver — (same as option 5)

Option 15: L0 Driver — (same as option 5)

Option 16: SI input — (same as option 4)

Option 17: SO Driver
- = 0: Standard push-pull output
- = 2: Open-drain output

Option 18: SK Driver — (same as option 17)

Option 19: IN0 Input — (same as option 9)

Option 20: IN3 Input — (same as option 9)

Option 21: G0 I/O Port — (same as option 17)

Option 22: G1 I/O Port — (same as option 17)

Option 23: G2 I/O Port — (same as option 17)

Option 24: G3 I/O Port — (same as option 17)

Option 25: D3 Output — (same as option 17)

Option 26: D2 Output — (same as option 17)

Option 27: D1 Output — (same as option 17)

Option List (Continued)

Option 28: D0 Output — (same as option 17)

Option 29: Internal Initialization Logic

= 0: Normal operation

= 1: No internal initialization logic

Option 30 = 0: No Option Available

Option 31: Timer

= 0: Time-base counter

= 1: External event counter

Option 32 = 0: No Option Available

Option 33: COP bonding. See note.

(1k and 2k Microcontroller)

= 0: 28-pin package

= 1: 24-pin package

(1k Microcontroller only)

= 3: 20-pin package

= 5: 24- and 20-pin package

Note:—If opt. #33 = 0 then opt. #9, 10, 19, and 20 must = 1.

If opt. #33 = 1 then opt. #9, 10, 19 and 20 must = 2, and option #31 must = 0.

If opt. #33 = 3 or 5 then opt. #9, 10, 19, 20 must = 2 and opt. #21, 22, 31 must = 0.

Option 34 = 0: No Option Available

Option 35 = 0: No Option Available

Option Table

The following option information is to be sent to National along with the EPROM.

	OPTION DATA			OPTION DATA	
OPTION 1 VALUE =	0	IS: GROUND PIN	OPTION 19 VALUE =		IS: IN0 INPUT
OPTION 2 VALUE =		IS: CKO PIN	OPTION 20 VALUE =		IS: IN3 INPUT
OPTION 3 VALUE =		IS: CKI INPUT	OPTION 21 VALUE =		IS: G0 I/O PORT
OPTION 4 VALUE =	1	IS: $\overline{\text{RESET}}$ INPUT	OPTION 22 VALUE =		IS: G1 I/O PORT
OPTION 5 VALUE =		IS: L7 DRIVER	OPTION 23 VALUE =		IS: G2 I/O PORT
OPTION 6 VALUE =		IS: L6 DRIVER	OPTION 24 VALUE =		IS: G3 I/O PORT
OPTION 7 VALUE =		IS: L5 DRIVER	OPTION 25 VALUE =		IS: D3 OUTPUT
OPTION 8 VALUE =		IS: L4 DRIVER	OPTION 26 VALUE =		IS: D2 OUTPUT
OPTION 9 VALUE =		IS: IN1 INPUT	OPTION 27 VALUE =		IS: D1 OUTPUT
OPTION 10 VALUE =		IS: IN2 INPUT	OPTION 28 VALUE =		IS: D0 OUTPUT
OPTION 11 VALUE =	0	IS: VCC PIN	OPTION 29 VALUE =		IS: INT INIT LOGIC
OPTION 12 VALUE =		IS: L3 DRIVER	OPTION 30 VALUE =	0	IS: N/A
OPTION 13 VALUE =		IS: L2 DRIVER	OPTION 31 VALUE =		IS: TIMER
OPTION 14 VALUE =		IS: L1 DRIVER	OPTION 32 VALUE =	0	IS: N/A
OPTION 15 VALUE =		IS: L0 DRIVER	OPTION 33 VALUE =		IS: COP BONDING
OPTION 16 VALUE =	1	IS: SI INPUT	OPTION 34 VALUE =	0	IS: N/A
OPTION 17 VALUE =		IS: SO DRIVER	OPTION 35 VALUE =	0	IS: N/A
OPTION 18 VALUE =		IS: SK DRIVER			

National Semiconductor

COP410C/COP411C/COP310C/COP311C Single-Chip CMOS Microcontrollers

General Description

The COP410C, COP411C, COP310C, and COP311C fully static, single-chip CMOS microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP411C is identical to the COP410C but with 16 I/O lines instead of 20. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller-oriented processor at a low end-product cost.

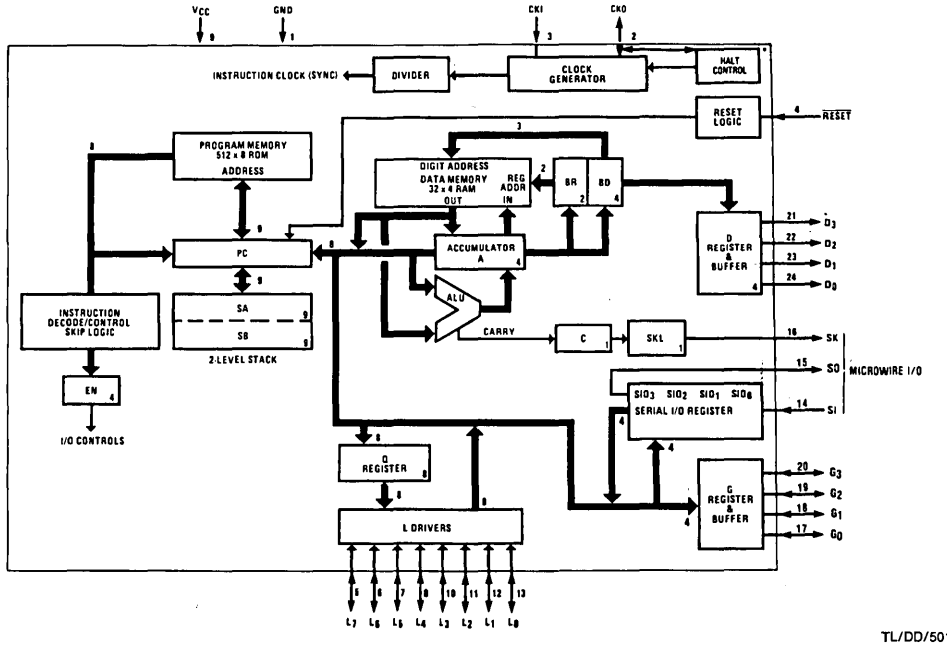
The COP310C/COP311C is the extended temperature range version of the COP410C/COP411C.

The COP404C should be used for exact emulation.

Features

- Lowest power dissipation (40 μW typical)
- Low cost
- Power-saving HALT Mode with Continue function
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 20 I/O lines (COP410C)
- Two-level subroutine stack
- DC to 4 μs instruction time
- Single supply operation (2.4V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of the COP400 family
- Extended temperature (−40°C to +85°C) devices available
- The military temperature range devices (−55°C to +125°C) are specified on COP210C/211C data sheet.

Block Diagram



TL/DD/5015-1

FIGURE 1. COP410C

COP410C/COP411C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	−0.3V to V_{CC}+0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple[5]			0.1 V_{CC}	V
Supply Current	V_{CC} = 2.4V, t_c = 125 μs V_{CC} = 5.0V, t_c = 16 μs V_{CC} = 5.0V, t_c = 4 μs (t_c is instruction cycle time)		80 500 2000	μA μA μA
HALT Mode Current[2]	V_{CC} = 5.0V, F_{IN} = 0 kHz V_{CC} = 2.4V, F_{IN} = 0 kHz		30 10	μA μA
Input Voltage Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		 0.9 V_{CC} 0.7 V_{CC}	 0.1 V_{CC} 0.2 V_{CC}	 V V V V
Hi-Z Input Leakage		−1	+1	μA
Input Capacitance			7	pF
Output Voltage Levels LSTTL Operation Logic High Logic Low CMOS Operation Logic High Logic Low	Standard Outputs V_{CC} = 5.0V ±10% I_{OH} = −25 μA I_{OL} = 400 μA I_{OH} = −10 μA I_{OL} = 10 μA	 2.7 V_{CC}−0.2 	 0.4 0.2	 V V V V
Output Current Levels[4] (Except CKO) Sink Source (Standard Option) Source (Low Current Option)	 V_{CC} = 4.5V, V_{OUT} = V_{CC} V_{CC} = 2.4V, V_{OUT} = V_{CC} V_{CC} = 4.5V, V_{OUT} = 0V V_{CC} = 2.4V, V_{OUT} = 0V V_{CC} = 4.5V, V_{OUT} = 0V V_{CC} = 2.4V, V_{OUT} = 0V	 1.2 0.2 −0.5 −0.1 −30 −6	 −330 −80	 mA mA mA mA μA μA
CKO Current Levels (As Clock Out) Sink ÷4 ÷8 ÷16 Source ÷4 ÷8 ÷16	 V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC} V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V 	 0.3 0.6 1.2 −0.3 −0.6 −1.2	 	 mA mA mA mA mA mA
Allowable Sink/Source Current Per Pin[4]			5	mA

COP410C/COP411C

DC Electrical Characteristics (Continued)

Parameter	Conditions	Min	Max	Units
Allowable Loading on CKO (as HALT I/O pin)			100	pF
Current Needed to Override HALT[3]				
To Continue	V_{CC} = 4.5V, V_{IN} = 0.2 V_{CC}		0.6	mA
To Halt	V_{CC} = 4.5V, V_{IN} = 0.7 V_{CC}		1.6	mA
TRI-STATE or Open Drain Leakage Current		−2	+2	μA

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating in the RC and crystal configurations.

Note 3: When forcing HALT, current is only needed for a short time (approximately 200 ns) to flip the HALT flip-flop.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 5: Voltage change must be less than 0.5V in a 1 ms period.

Note 6: This parameter is only sampled and not 100% tested.

Note 7: Variation due to the device included.

COP410C/COP411C

AC Electrical Characteristics $0°C \le T_A \le 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)	$V_{CC} \ge 4.5V$	4	DC	μs
	$4.5V > V_{CC} \ge 2.4V$	16	DC	μs
Operating CKI Frequency ÷4 mode	$V_{CC} \ge 4.5V$	DC	1.0	MHz
÷8 mode	$V_{CC} \ge 4.5V$	DC	2.0	MHz
÷16 mode	$V_{CC} \ge 4.5V$	DC	4.0	MHz
÷4 mode	$4.5V > V_{CC} \ge 2.4V$	DC	250	kHz
÷8 mode	$4.5V > V_{CC} \ge 2.4V$	DC	500	kHz
÷16 mode	$4.5V > V_{CC} \ge 2.4V$	DC	1.0	MHz
Instruction Cycle Time RC Oscillator[7]	R = 30k ± 5%, V_{CC} = 5V, C = 82 pF ± 5% (÷4 Mode)	8	16	μs
Duty Cycle[6]	f_I = 4 MHz	40	60	%
Rise Time[6]	f_I = 4 MHz External Clock		60	ns
Fall Time[6]	f_I = 4 MHz External Clock		40	ns
Inputs (See *Figure 3*)				
t_{SETUP} G Inputs	$V_{CC} \ge 4.5V$	tc/4+0.7		μs
t_{SETUP} SI Input	$V_{CC} \ge 4.5V$	0.3		μs
t_{SETUP} All Others	$V_{CC} \ge 4.5V$	1.7		μs
t_{HOLD}	$V_{CC} \ge 4.5V$	0.25		μs
	$V_{CC} \ge 2.4V$	1.0		μs
Output Propagation Delay	V_{OUT} = 1.5V, C_L = 100 pF, R_L = 5k			
t_{PD1}, t_{PD0}	$V_{CC} \le 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$V_{CC} \le 2.4V$		4.0	μs

COP310C/COP311C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	−0.3V to V_{CC}+0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		3.0	5.5V	V
Power Supply Ripple[5]			0.1 V_{CC}	V
Supply Current	V_{CC} = 3.0V, t_c = 125 μs V_{CC} = 5.0V, t_c = 16 μs V_{CC} = 5.0V, t_c = 4 μs (t_c is instruction cycle time)		100 600 2500	μA μA μA
HALT Mode Current[2]	V_{CC} = 5.0V, F_{IN} = 0 kHz V_{CC} = 3.0V, F_{IN} = 0 kHz		50 20	μA μA
Input Voltage Levels $\overline{RESET}$, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		 0.9 V_{CC} 0.7 V_{CC}	 0.1 V_{CC} 0.2 V_{CC}	 V V V V
Hi-Z Input Leakage		−2	+2	μA
Input Capacitance			7	pF
Output Voltage Levels LSTTL Operation Logic High Logic Low CMOS Operation Logic High Logic Low	Standard Outputs V_{CC} = 5.0V ±10% I_{OH} = −25 μA I_{OL} = 400 μA I_{OH} = −10 μA I_{OL} = 10 μA	 2.7 V_{CC}−0.2 	 0.4 0.2	 V V V V
Output Current Levels[4] (Except CKO) Sink Source (Standard Option) Source (Low Current Option)	 V_{CC} = 4.5V, V_{OUT} = V_{CC} V_{CC} = 3.0V, V_{OUT} = V_{CC} V_{CC} = 4.5V, V_{OUT} = 0V V_{CC} = 3.0V, V_{OUT} = 0V V_{CC} = 4.5V, V_{OUT} = 0V V_{CC} = 3.0V, V_{OUT} = 0V	 1.2 0.2 −0.5 −0.1 −30 −8	 −440 −200	 mA mA mA mA μA μA
CKO Current Levels (As Clock Out) Sink ÷4 ÷8 ÷16 Source ÷4 ÷8 ÷16	 V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC} V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V 	 0.3 0.6 1.2 −0.3 −0.6 −1.2	 	 mA mA mA mA mA mA
Allowable Sink/Source Current Per Pin[4]			5	mA

COP310C/COP311C

DC Electrical Characteristics (Continued)

Parameter	Conditions	Min	Max	Units
Allowable Loading on CKO (as HALT I/O pin)			100	pF
Current Needed to Override HALT[3]				
To Continue	V_{CC} = 4.5V, V_{IN} = 0.2 V_{CC}		0.8	mA
To Halt	V_{CC} = 4.5V, V_{IN} = 0.7 V_{CC}		2.0	mA
TRI-STATE or Open Drain Leakage Current		−4	+4	μA

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating in the RC and crystal configurations.

Note 3: When forcing HALT, current is only needed for a short time (approximately 200 ns) to flip the HALT flip-flop.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 5: Voltage change must be less than 0.5V in a 1 ms period.

Note 6: This parameter is only sampled and not 100% tested.

Note 7: Variation due to the device included.

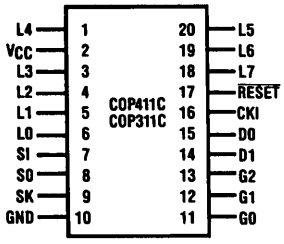
COP310C/COP311C

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)	$V_{CC} \ge 4.5V$	4	DC	μs
	4.5V > $V_{CC} \ge$ 3.0V	16	DC	μs
Operating CKI ÷4 mode	$V_{CC} \ge 4.5V$	DC	1.0	MHz
Frequency ÷8 mode	$V_{CC} \ge 4.5V$	DC	2.0	MHz
÷16 mode	$V_{CC} \ge 4.5V$	DC	4.0	MHz
÷4 mode	4.5V > $V_{CC} \ge$ 3.0V	DC	250	kHz
÷8 mode	4.5V > $V_{CC} \ge$ 3.0V	DC	500	kHz
÷16 mode	4.5V > $V_{CC} \ge$ 3.0V	DC	1.0	MHz
Instruction Cycle Time RC Oscillator[7]	R = 30k ±5%, V_{CC} = 5V C = 82 pF ±5% (÷4 Mode)	8	16	μs
Duty Cycle[6]	f_I = 4 MHz	40	60	%
Rise Time[6]	f_I = 4 MHz External Clock		60	ns
Fall Time[6]	f_I = 4 MHz External Clock		40	ns
Inputs (See *Figure 3*)				
t_{SETUP} G Inputs	$V_{CC} \ge 4.5V$	tc/4+0.7		μs
t_{SETUP} SI Input	$V_{CC} \ge 4.5V$	0.3		μs
t_{SETUP} All Others	$V_{CC} \ge 4.5V$	1.7		μs
t_{HOLD}	$V_{CC} \ge 4.5V$	0.25		μs
	$V_{CC} \ge 3.0V$	1.0		μs
Output Propagation Delay	V_{OUT} = 1.5V, C_L = 100 pF, R_L = 5k			
t_{PD1}, t_{PD0}	$V_{CC} \le 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$V_{CC} \le 3.0V$		4.0	μs

Connection Diagrams

S.O. Wide and DIP



TL/DD/5015-2

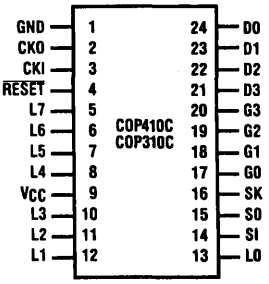
Top View

Order Number COP311C-XXX/D or COP411C-XXX/D
See NS Hermetic Package Number D20A

Order Number COP311C-XXX/N or COP411C-XXX/N
See NS Molded Package Number N20A

Order Number COP311C-XXX/WM or COP411C-XXX/WM
See NS Surface Mount Package Number M20B

S.O. Wide and DIP



TL/DD/5015-3

Top View

Order Number COP310C-XXX/D or COP410C-XXX/D
See NS Hermetic Package Number D24C

Order Number COP310C-XXX/N or COP410C-XXX/N
See NS Molded Package Number N24A

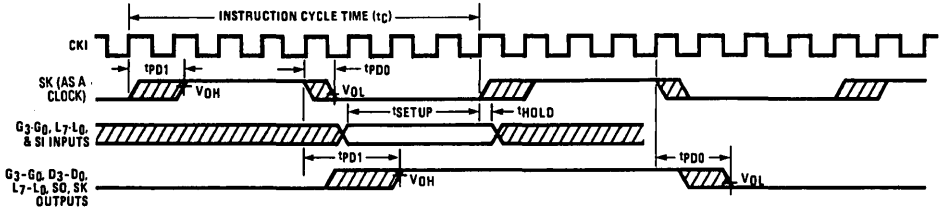
Order Number COP310C-XXX/WM or COP410C-XXX/WM
See NS Surface Mount Package Number M24B

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L_7–L_0	8-bit bidirectional I/O port with TRI-STATE	SK	Logic-controlled clock (or general purpose output)
G_3–G_0	4-bit bidirectional I/O port (G_2–G_0 for 20-pin package)	CKI	System oscillator input
D_3–D_0	4-bit general purpose output port (D_1–D_0 for 20-pin package)	CKO	Crystal oscillator output, or HALT mode I/O port (24-pin package only)
SI	Serial input (or counter input)	$\overline{\text{RESET}}$	System reset input
SO	Serial output (or general purpose output)	V_{CC}	System power supply
		GND	System Ground

Timing Diagram



TL/DD/5015-4

FIGURE 3. Input/Output (Divide-by-8 Mode)

Functional Description

To ease reading of this description, only COP410C and/or COP411C are referenced; however, all such references apply equally to COP310C and/or COP311C, respectively.

A block diagram of the COP410C is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP410C/411C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8 × 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4.* The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP410C/411C is designed to ensure fully static operation of the device.

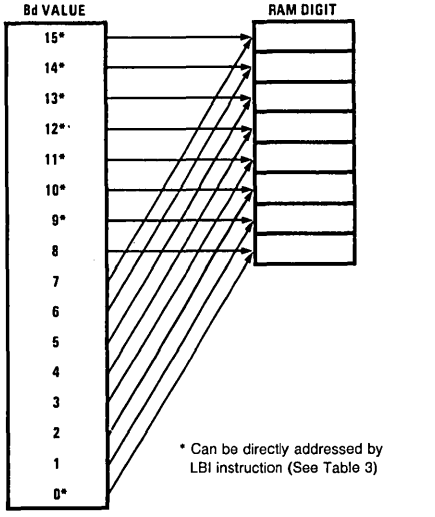
The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP410C/411C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.



TL/DD/5015–5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP410C/411C is MICROWIRE compatible.

The D register provides four general purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.
2. EN 1 is not used, it has no effect on the COP410C/411C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

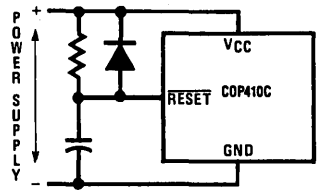
INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

When V_{CC} power is applied, the internal reset logic will keep the chip in initialization mode for up to 2500 instruction cycles. If the CKI clock is running at a low frequency, this could take a long time, therefore, the internal logic should be disabled by a mask option with initialization controlled solely by $\overline{\text{RESET}}$ pin.

Note: If CKI clock is less than 32 kHz, the internal reset logic (Option 25 = 1) *must* be disabled and the external RC network *must* be present.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/5015–6

RC > 5 × Power Supply Rise Time
and RC > 100 × CKI Period

FIGURE 5. Power-Up Clear Circuit

COP411C

If the COP410C is bonded as a 20-pin package, it becomes the COP411C, illustrated in *Figure 2,* COP410C/411C Connection Diagrams. Note that the COP411C does not contain D2, D3, G3, or CKO. Use of this option, of course, precludes use of D2, D3, G3, and CKO options. All other options are available for the COP411C.

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

HALT MODE

The COP410C/411C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip also may be halted by the HALT instruction or by forcing CKO high when it is used as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode has slight differences depending upon the type of oscillator used.

a. 1-pin oscillator—RC or external

The HALT mode may be entered into by either program control (HALT instruction) or by forcing CKO to a logic "1" state.

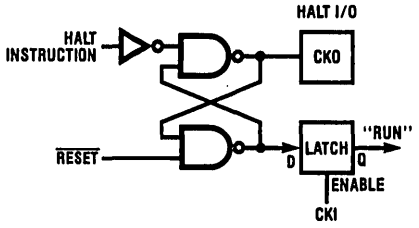
The circuit may be awakened by one of two different methods:

1) Continue function. By forcing CKO to a logic "0", the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.

2) Restart. Forcing the $\overline{\text{RESET}}$ pin to a logic "0" will restart the chip regardless of HALT or CKO (see initialization).

b. 2-pin oscillator—crystal

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.



TL/DD/5015-7

Halt I/O Port

CKO Pin Options

In a crystal-controlled oscillator system, CKO is used as an output to the crystal network. CKO will be forced high during the execution of a HALT instruction, thus inhibiting the crystal network. If a 1-pin oscillator system is chosen (RC or external), CKO will be selected as HALT and is an I/O flip-flop which is an indicator of the HALT status. An external signal can override this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and the CKO output will go high to keep the chip stopped. By forcing a low level to CKO, the chip will continue and CKO output will go low.

All features associated with the CKO I/O pin are available with the 24-pin package only.

OSCILLATOR OPTIONS

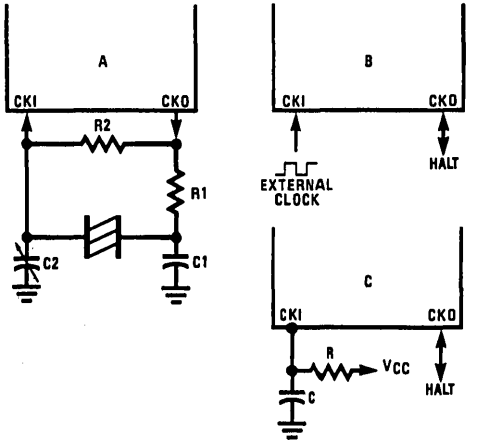
There are three options available that define the use of CKI and CKO.

a. Crystal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optionally by 8 or 4).

b. External Oscillator. CKI is configured as LSTTL-compatible input accepting an external clock signal. The external frequency is divided by 16 (optionally by 8 or 4) to give the instruction cycle time. CKO is the HALT I/O port.

c. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.



TL/DD/5015-8

FIGURE 6. COP410C Oscillator

Crystal or Resonator

Crystal Value	Component Value R1	R2	C1 pF	C2 pF
32 kHz	220k	20M	30	5-36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6-36
4.0 MHz	1k	1M	30	6-36

RC-Controlled Oscillator

R	C	Cycle Time	V_{CC}
15k	82 pF	4-9 μs	≥4.5V
30k	82 pF	8-16 μs	≥4.5V
47k	100 pF	16-32 μs	2.4 to 4.5
Note: 15k ≤ R ≤ 150k, 50 pf ≤ C ≤ 150 pF			

COP410C/COP411C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410C/411C instruction set.

TABLE II. COP410C/411C Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP410C/411C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5−	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP410C/411C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	0110 | 000 | a_8	a → PC	None	Jump
		–	$a_{7:0}$			
JP	a	–	1 | $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 1)
			or			
		–	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	10 | $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 2)
JSR	a	6–	0110 | 100 | a_8	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
		–	$a_{7:0}$			
RET		48	0100 | 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 10011	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33	0011 | 0011		None	Halt processor
		38	0011 | 1000			
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 | 0011	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 | 1100	RAM(B) → $Q_{3:0}$		
CQMA		33	0011 | 0011	$Q_{7:4}$ → RAM(B)	None	Copy Q to RAM, A
		2C	0010 | 1100	$Q_{3:0}$ → A		
LD	r	–5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8,A,M) → Q SA → SB	None	Load Q Indirect
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23	0010 | 0011	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
		BF	1011 | 1111			

Instruction Set (Continued)

TABLE III. COP410C/411C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	−7	00 | r | 0111	RAM(B) ⟷ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	−4	00 | r | 0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	–	00 | r | (d - 1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6–	0011 | 0011 0010 | y	y → EN	None	Load EN Immediate
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 | 0011 0010 | 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0 = 0$	
	1	11	0001 | 0001	2nd byte	$G_1 = 0$	
	2	03	0000 | 0011	2nd byte	$G_2 = 0$	
	3	13	0010 | 0011	2nd byte	$G_3 = 0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1 = 0$	
	2	03	0000 | 0011		$RAM(B)_2 = 0$	
	3	13	0001 | 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 | 0011 0010 | 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 | 0011 0010 | 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 | 0011 0011 | 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP410C/411C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register). If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 $\rightarrow$ SA $\rightarrow$ SB) and replaces the least significant eight bits of the PC as follows: A $\rightarrow$ $PC_{7:4}$, RAM(B) $\rightarrow$ $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB $\rightarrow$ SA $\rightarrow$ PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA $\rightarrow$ SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

INSTRUCTION SET NOTES

a. The first word of a COP410C/411C program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).

c. The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw additional current. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP410C current drain.

$$Ic = Iq + (V \times 20 \times Fi) + (V \times 1280 \times Fl/Dv)$$

where Ic = chip current drain in microamps

Iq = quiescent leakage current (from curve)

Fl = CKI frequency in megahertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 4),

$$Ic = 10 + (5 \times 20 \times 0.4) + (5 \times 1280 \times 0.4/4)$$

$$Ic = 10 + 40 + 640 = 690\ \mu A$$

I/O OPTIONS

COP410C/411C outputs have the following optional configurations, illustrated in *Figure 7*:

a. Standard. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

b. Low Current. This is the same configuration as (a) above except that the sourcing current is much less.

c. Open Drain. An N-channel device to ground only, allowing external pull-up as required by the user's application.

d. Standard TRI-STATE L Output. A CMOS output buffer similar to (a) which may be disabled by program control.

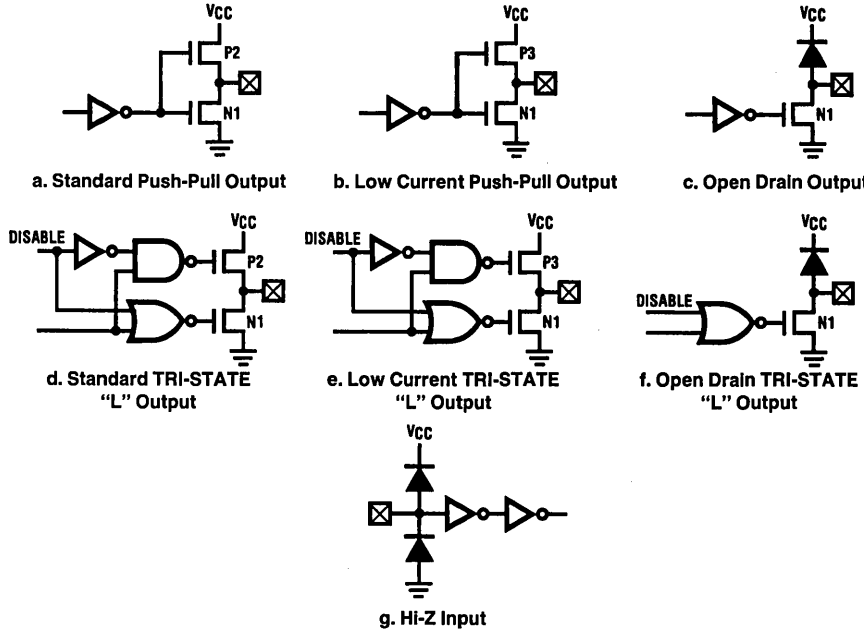
e. Low-Current TRI-STATE L Output. This is the same as (d) above except that the sourcing current is much less.

f. Open-Drain TRI-STATE L Output. This has the N-channel device to ground only.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (*Figure 7g*).

When using either the G or L I/O ports as inputs, a pull-up device is necessary. This can be an external device or the following alternative is available: Select the low-current output option. Now, by setting the output registers to a logic "1" level, the P-channel devices will act as the pull-up load. Note that when using the L ports in this fashion, the Q registers must be set to a logic "1" level and the L drivers *must be enabled* by an LEI instruction.

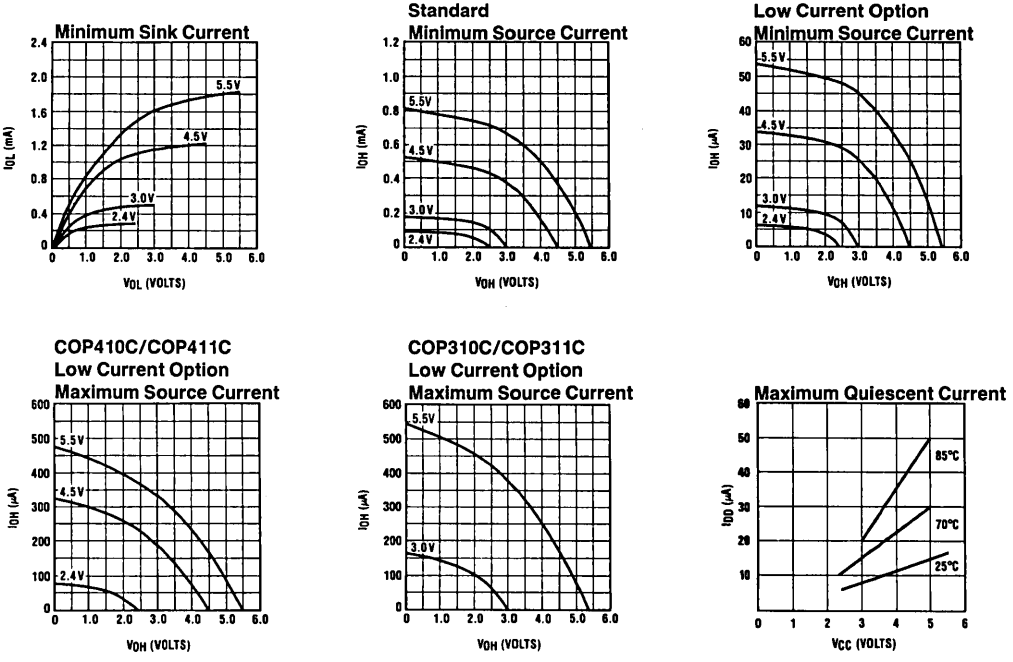
Functional Description (Continued)



TL/DD/5015–9

FIGURE 7. I/O Configurations

Typical Performance Characteristics



TL/DD/5015–10

FIGURE 8

All output drivers uses one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.

Option List

The COP410C/411C mask-programmable options are assigned numbers which correspond with the COP410C pins.

The following is a list of COP410C options. When specifying a COP411 chip, options 20, 21, and 22 must be set to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: 0 = Ground Pin. No options available.

Option 2: CKO I/O Port. (Determined by Option 3.)
= 0: No option.
(a. is crystal oscillator output for two pin oscillator.
b. is HALT I/O for one pin oscillator.)

Option 3: CKI Input.
= 0: Crystal-controlled oscillator input (÷ 4).
= 1: Single-pin RC-controlled oscillator (÷ 4).
= 2: External oscillator input (÷ 4).
= 3: Crystal oscillator input (÷ 8).
= 4: External oscillator input (÷ 8).
= 5: Crystal oscillator input (÷ 16).
= 6: External oscillator input (÷ 16).

Option 4: $\overline{\text{RESET}}$ Input = 1: Hi-Z input. No option available.

Option 5: L_7 Driver
= 0: Standard TRI-STATE push-pull output.
= 1: Low-current TRI-STATE push-pull output.
= 2: Open-drain TRI-STATE output.

Option 6: L_6 Driver. (Same as Option 5.)

Option 7: L_5 Driver. (Same as Option 5.)

Option 8: L_4 Driver. (Same as Option 5.)

Option 9: V_{CC} Pin = 0 no option.

Option 10: L_3 Driver. (Same as Option 5.)

Option 11: L_2 Driver. (Same as Option 5.)

Option 12: L_1 Driver. (Same as Option 5.)

Option 13: L_0 Driver. (Same as Option 5.)

Option 14: SI Input.
No option available.
= 1: Hi-Z input.

Option 15: SO Output.
= 0: Standard push-pull output.
= 1: Low-current push-pull output.
= 2: Open-drain output.

Option 16: SK Driver. (Same as Option 15.)

Option 17: G_0 I/O Port. (Same as Option 15.)

Option 18: G_1 I/O Port. (Same as Option 15.)

Option 19: G_2 I/O Port. (Same as Option 15.)

Option 20: G_3 I/O Port. (Same as Option 15.)

Option 21: D_3 Output. (Same as Option 15.)

Option 22: D_2 Output. (Same as Option 15.)

Option 23: D_1 Output. (Same as Option 15.)

Option 24: D_0 Output. (Same as Option 15.)

Option 25: Internal Initialization Logic.
= 0: Normal operation.
= 1: No internal initialization logic.

Option 26: No option available.

Option 27: COP Bonding
= 0: COP410C (24-pin device).
= 1: COP411C (20-pin device). See note.
= 2: COP410C and COP411C. See note.

Note: If opt. #27 = 1 or 2 then opt #20 must = 0.

Option Table

Please fill out a photocopy of the option table and send it along with your EPROM.

Option Table

Option 1 Value =	0	is: Ground Pin	
Option 2 Value =	0	is: CKO Pin	
Option 3 Value =		is: CKI Input	
Option 4 Value =	1	is: $\overline{\text{RESET}}$ Input	
Option 5 Value =		is: L_7 Driver	
Option 6 Value =		is: L_6 Driver	
Option 7 Value =		is: L_5 Driver	
Option 8 Value =		is: L_4 Driver	
Option 9 Value =	0	is: V_{CC} Pin	
Option 10 Value =		is: L_3 Driver	
Option 11 Value =		is: L_2 Driver	
Option 12 Value =		is: L_1 Driver	
Option 13 Value =		is: L_0 Driver	
Option 14 Value =	1	is: SI Input	
Option 15 Value =		is: SO Output	
Option 16 Value =		is: SK Driver	
Option 17 Value =		is: G_0 I/O Port	
Option 18 Value =		is: G_1 I/O Port	
Option 19 Value =		is: G_2 I/O Port	
Option 20 Value =		is: G_3 I/O Port	
Option 21 Value =		is: D_3 Output	
Option 22 Value =		is: D_2 Output	
Option 23 Value =		is: D_1 Output	
Option 24 Value =		is: D_0 Output	
Option 25 Value =		is: Internal Initialization Logic	
Option 26 Value =	0	is: N/A	
Option 27 Value =		is: COP Bonding	

National Semiconductor

COP410L/COP411L/COP310L/COP311L Single-Chip N-Channel Microcontrollers

General Description

The COP410L and COP411L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP411L is identical to the COP410L, but with 16 I/O lines instead of 19. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

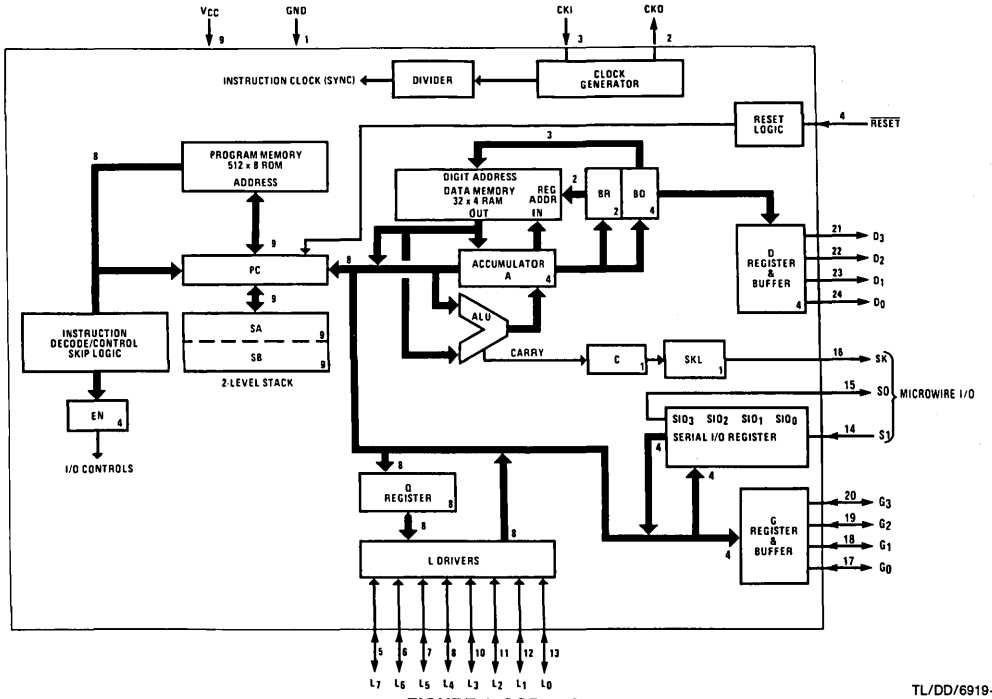
The COP310L and COP311L are exact functional equivalents but extended temperature versions of COP410L and COP411L respectively.

The COP401L should be used for exact emulation.

Features

- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 19 I/O lines (COP410L)
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device — COP310L/COP311L (−40°C to +85°C)
- Wider supply range (4.5V–9.5V) optionally available

Block Diagram



TL/DD/6919–1

FIGURE 1. COP410L

COP410L/COP411L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	
COP410L	0.75W at 25°C 0.4W at 70°C
COP411L	0.65W at 25°C 0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Optional Operating Voltage (V_{CC})		4.5	9.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 3)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

Note 3: TRI-STATE® and LED configurations are excluded.

COP410L/COP411L

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	1.8		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	1.2		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.9		mA
L_0–L_7 Outputs, G_0–G_3 and LSTTL D_0–D_3 Outputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
D_0–D_3 Outputs with High Current Options (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 1.0V$	15		mA
	$V_{CC} = 6.3V, V_{OL} = 1.0V$	11		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7.5		mA
D_0–D_3 Outputs with Very High Current Options (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 1.0V$	30		mA
	$V_{CC} = 6.3V, V_{OL} = 1.0V$	22		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	15		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−140	−800	μA
	$V_{CC} = 6.3V, V_{OH} = 2.0V$	−75	−480	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−30	−250	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 4.75V$	−1.4		mA
	$V_{CC} = 6.3V, V_{OH} = 2.4V$	−1.4		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−1.5	−18	mA
	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.5	−13	mA
LED Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−3.0	−35	mA
	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−3.0	−25	mA
TRI-STATE Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 5.5V$	−0.75		mA
	$V_{CC} = 6.3V, V_{OH} = 3.2V$	−0.8		mA
	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 5.5V$	−1.5		mA
	$V_{CC} = 6.3V, V_{OH} = 3.2V$	−1.6		mA
	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−140	μA
CKO Output				
RAM Power Supply Option Power Requirement	$V_R = 3.3V$		1.5	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pin			1.5	mA

COP310L/COP311L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	
COP310L	0.75W at 25°C 0.25W at 85°C
COP311L	0.65W at 25°C 0.20W at 85°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Optional Operating Voltage (V_{CC})		4.5	7.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8) Crystal Input				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 3)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

Note 3: TRI-STATE and LED configurations are excluded.

COP310L/COP311L

DC Electrical Characteristics (Continued)

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless othewise noted

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 7.5V, V_{OL} = 0.4V$	1.4		mA
	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0-L_7 Outputs, G_0-G_3 and LSTTL D_0-D_3 Outputs (I_{OL})	$V_{CC} = 7.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
D_0-D_3 Outputs with High Current Options (I_{OL})	$V_{CC} = 7.5V, V_{OL} = 1.0V$	12		mA
	$V_{CC} = 5.5V, V_{OL} = 1.0V$	9		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7		mA
D_0-D_3 Outputs with Very High Current Options (I_{OL})	$V_{CC} = 7.5V, V_{OL} = 1.0V$	24		mA
	$V_{CC} = 5.5V, V_{OL} = 1.0V$	18		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	14		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	1.5		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−100	−900	μA
	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	$V_{CC} = 7.5V, V_{OH} = 3.75V$	−0.85		mA
	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0-L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−1.4	−27	mA
	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−0.7	−15	μA
LED Configuration, L_0-L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−2.7	−54	mA
	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.4	−30	μA
TRI-STATE Configuration, L_0-L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 7.5V, V_{OH} = 4.0V$	−0.7		mA
	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−0.6		mA
	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration, L_0-L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 7.5V, V_{OH} = 4.0V$	−1.4		mA
	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−1.2		mA
	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−200	μA
CKO Output RAM Power Supply Option Power Requirement	$V_R = 3.3V$		2.0	mA
TRI-STATE Output Leakage Current		−5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7-L_4, G Port			4	mA
L_3-L_0			4	mA
Any Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7-L_4			60	mA
L_3-L_0			60	mA
Each L Pin			25	mA
Any Other Pins			1.5	mA

AC Electrical Characteristics

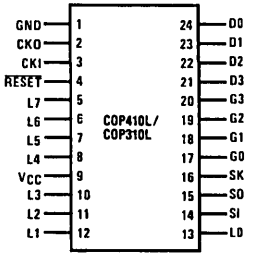
COP410L/411L: $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted
COP310L/311L: $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time — t_C		16	40	μs
CKI				
Input Frequency — f_I	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 0.5 MHz		500	ns
Fall Time			200	ns
CKI Using RC (÷4) (Note 1)	R = 56 kΩ ±5% C = 100 pF ±10%			
Instruction Cycle Time		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
INPUTS				
G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

Note 1: Variation due to the device included.

Connection Diagrams

SO Wide and DIP



TL/DD/6919-2

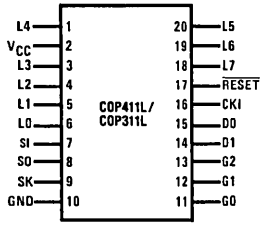
Top View

Order Number COP310L-XXX/D or COP410L-XXX/D
See NS Hermetic Package Number D24C

Order Number COP310L-XXX/N or COP410L-XXX/N
See NS Molded Package Number N24A

Order Number COP310L-XXX/WM or COP410L-XXX/WM
See NS Surface Mount Package Number M24B

SO Wide and DIP



TL/DD/6919-3

Top View

Order Number COP311L-XXX/D or COP411L-XXX/D
See NS Hermetic Package Number D24C

Order Number COP311L-XXX/N or COP411L-XXX/N
See NS Molded Package Number N20A

Order Number COP311L-XXX/WM or COP411L-XXX/WM
See NS Surface Mount Package Number M24B

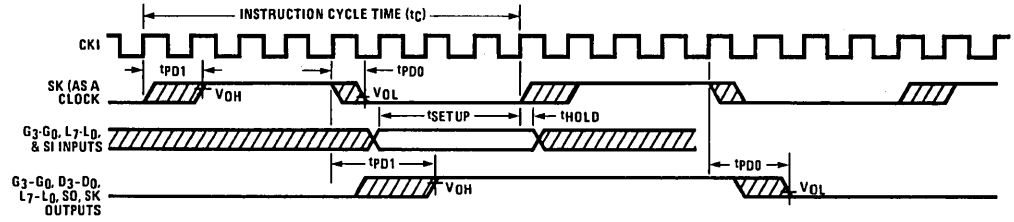
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports (G_2–G_0 for COP411L)
D_3–D_0	4 general purpose outputs (D_1–D_0 for COP411L)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)

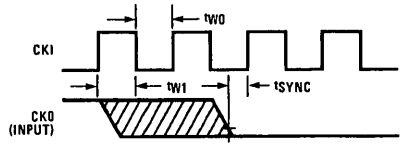
Pin	Description
CKI	System oscillator input
CKO	System oscillator output (or RAM power supply or SYNC input) (COP410L only)
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams



TL/DD/6919-4

FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)



TL/DD/6919-5

FIGURE 3a. Synchronization Timing

Functional Description

A block diagram of the COP410L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP410L/COP411L also apply to the COP310L/COP311L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP410L/411L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

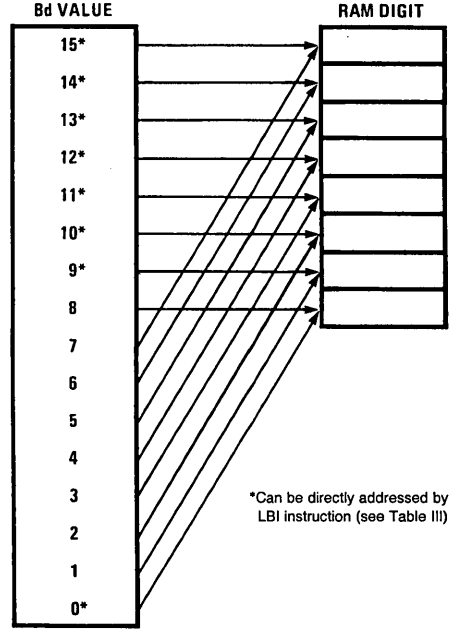
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).



TL/DD/6919-6

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP410L/411L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

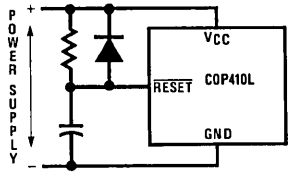
The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP410L/COP411L operation.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below *(Figure 5)*. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



RC ≥ 5 × Power Supply Rise Time TL/DD/6919–7

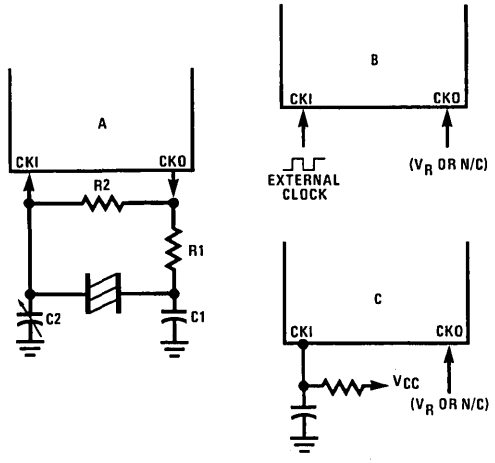
FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/6919–8

Ceramic Resonator Oscillator

Resonator Value	Components Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time In μs
51	100	19 ±15%
82	56	19 ±13%

Note: 200 kΩ ≥ R ≥ 25 kΩ. 360 pF ≥ C ≥ 50 pF. Does not include tolerances.

FIGURE 6. COP410L/411L Oscillator

OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 6*.

a. Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8. This is not available in the COP411L.

b. External Oscillator. CKI is an external clock input signal. The external frequency is divided by 4 to give the instruction frequency time. CKO is now available to be used as the RAM power supply (V_R), or no connection.

Note: No CKO on COP411L.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or no connection.

CKO PIN OPTIONS

In a resonator controlled oscillator system, CKO is used as an output to the resonator network. As an option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using no connection option is appropriate in applications where the COP410L system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. During normal operation, V_R must be within the operating range of the chip with $(V_{CC} - 1) \leq V_R \leq V_{CC}$.
3. V_R must be ≥ 3.3V with V_{CC} off.

I/O OPTIONS

COP410L/411L inputs and outputs have the following optional configurations, illustrated in *Figure 7*:

a. Standard—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. Standard L—same as **a.**, but may be disabled. Available on L outputs only.

e. Open Drain L—same as **b.**, but may be disabled. Available on L outputs only.

f. LED Direct Drive—an enhancement mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note: Series current limiting resistors must be used if LEDs are driven directly and higher operating voltage option is selected.

g. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

Functional Description (Continued)

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP410L/411L system.

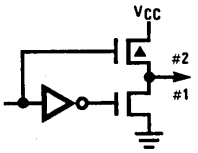
The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in **d.**, **e.**, **f.**, or **g.**

An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 8*, device 2.) However, when the L port is used as input, the disabled depletion device CANNOT be relied on to source sufficient current to pull an input to a logic "1".

COP411L

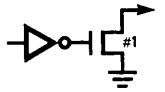
If the COP410L is bonded as a 20-pin device, it becomes the COP411L, illustrated in *Figure 2*, COP410L/411L Connection Diagrams. Note that the COP411L does not contain D2, D3, G3, or CKO. Use of this option of course precludes use of D2, D3, G3, and CKO options. All other options are available for the COP411L.

a. Standard Output



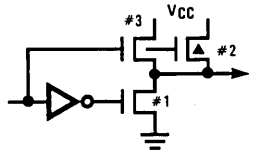
TL/DD/6919-9

b. Open-Drain Output



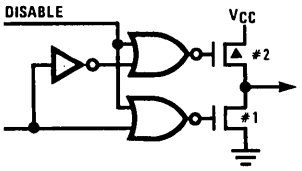
TL/DD/6919-10

c. Push-Pull Output



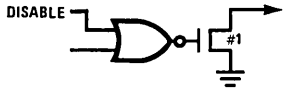
TL/DD/6919-11

d. Standard L Output



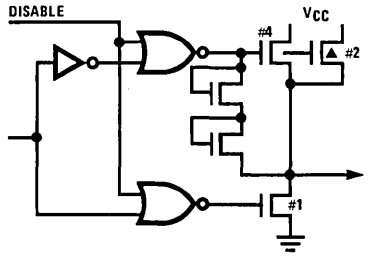
TL/DD/6919-12

e. Open-Drain L Output



TL/DD/6919-13

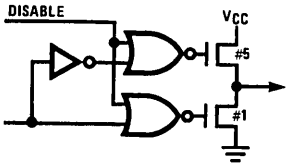
f. LED (L Output)



(▲ is depletion device)

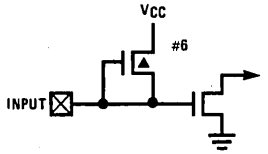
TL/DD/6919-14

g. TRI-STATE Push-Pull (L Output)



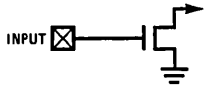
TL/DD/6919-15

h. Input with Load



TL/DD/6919-16

i. Hi-Z Input

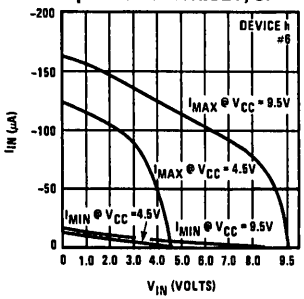


TL/DD/6919-17

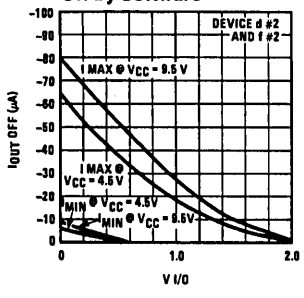
FIGURE 7. Input and Output Configurations

Typical Performance Characteristics

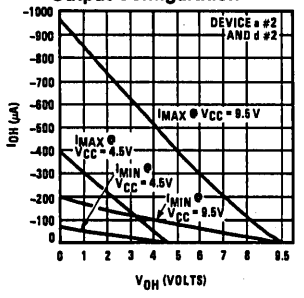
Input Current $\overline{\text{RESET}}$, SI



Input Current for L0 through L7 when Output Programmed Off by Software

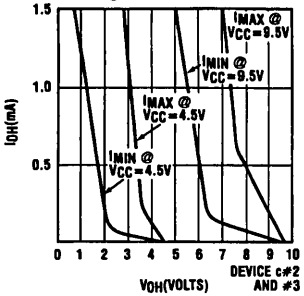


Source Current for Standard Output Configuration

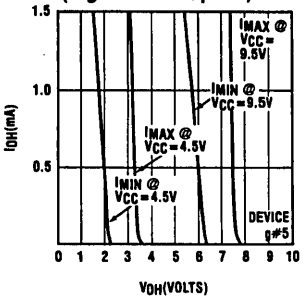


TL/DD/6919-18

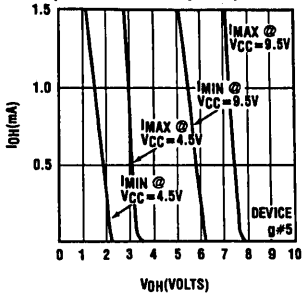
Source Current for SO and SK in Push-Pull Configuration



Source Current for L0 through L7 in TRI-STATE Configuration (High Current Option)

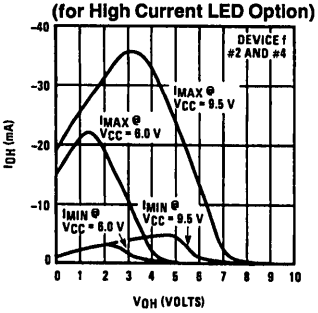


Source Current for L0 through L7 in TRI-STATE Configuration (Low Current Option)

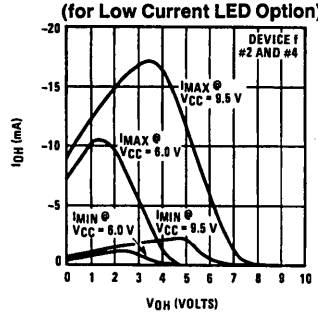


TL/DD/6919-19

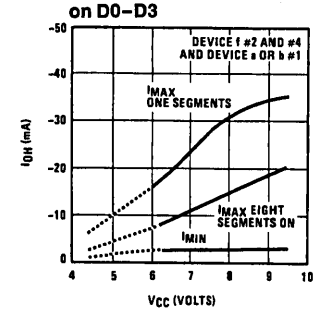
LED Output Source Current (for High Current LED Option)



LED Output Source Current (for Low Current LED Option)



LED Output Direct Segment and Direct Drive High Current Options on L0–L7 Very High Current Options on D0–D3

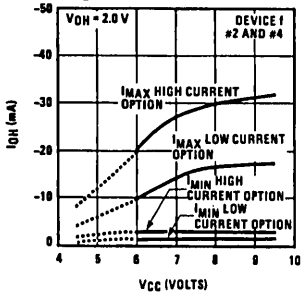


TL/DD/6919-20

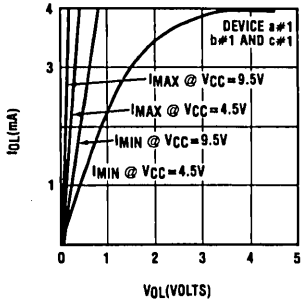
FIGURE 8a. COP410L/COP411L I/O DC Current Characteristics

Typical Performance Characteristics (Continued)

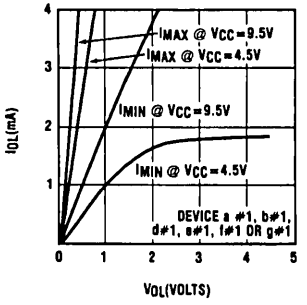
LED Output Direct Segment Drive



Output Sink Current for SO and SK

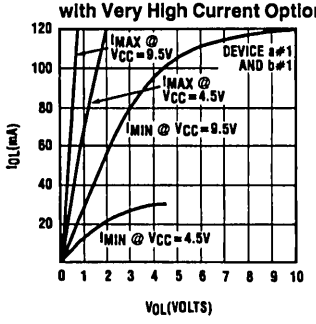


Output Sink Current for L0–L7 and Standard Drive Option for D0–D3 and G0–G3

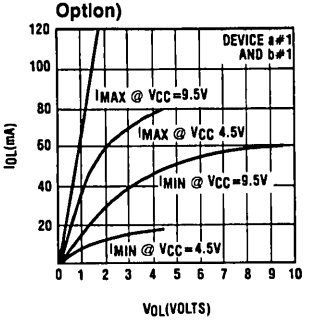


TL/DD/6919-21

Output Sink Current for D0–D3 with Very High Current Option



Output Sink Current for D0–D3 (for High Current Option)



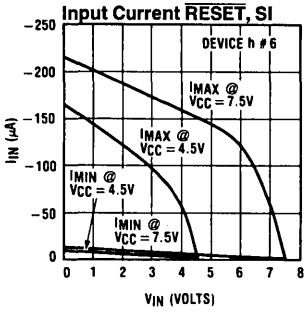
TL/DD/6919-22

FIGURE 8a. COP410L/COP411L I/O DC Current Characteristics (Continued)

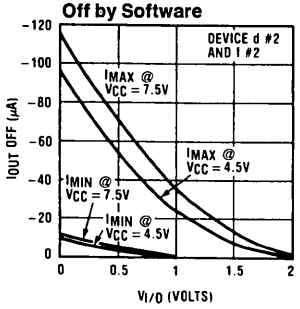
1

Typical Performance Characteristics (Continued)

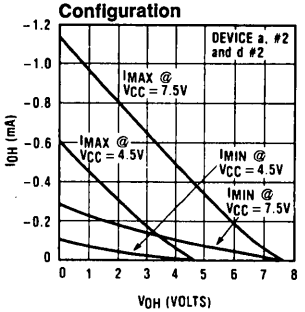
Input Current $\overline{\text{RESET}}$, SI



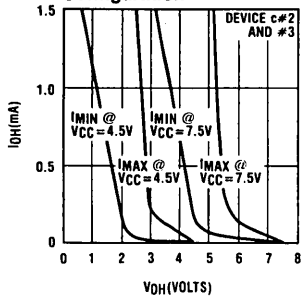
Input Current for L0–L7 when Output Programmed Off by Software



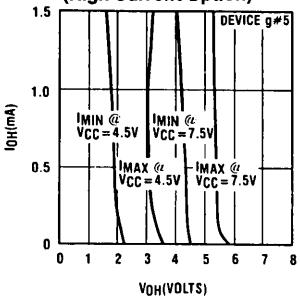
Source Current for Standard Output Configuration



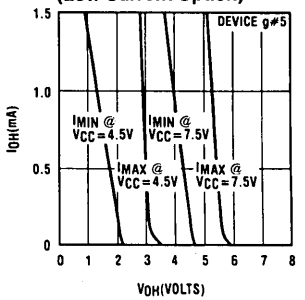
Source Current for SO and SK in Push-Pull Configuration



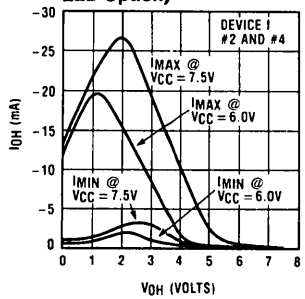
Source Current for L0–L7 in TRI-STATE Configuration (High Current Option)



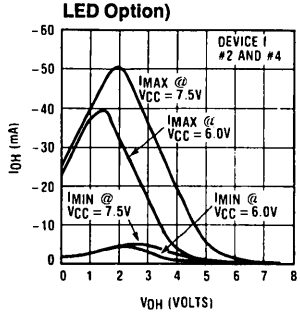
Source Current for L0–L7 in TRI-STATE Configuration (Low Current Option)



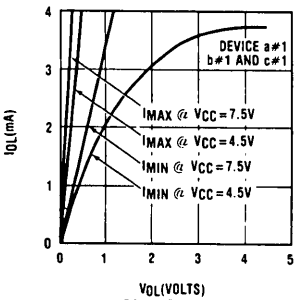
LED Output Source Current (for Low Current LED Option)



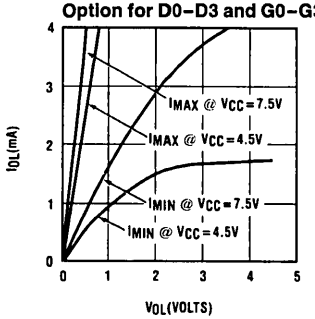
LED Output Source Current (for High Current LED Option)



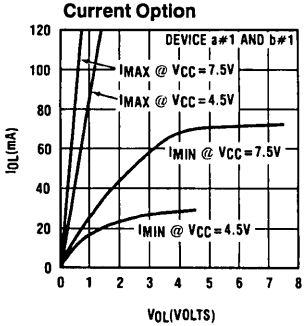
Output Sink Current for SO and SK



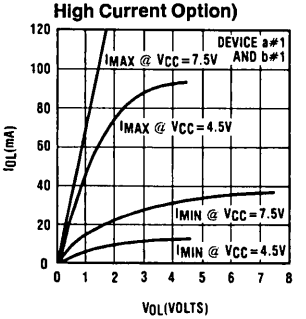
Output Sink Current for L0–L7 and Standard Drive Option for D0–D3 and G0–G3



Output Sink Current for D0–D3 with Very High Current Option



Output Sink Current for D0–D3 (for High Current Option)



TL/DD/6919–23

FIGURE 8b. COP310L/COP311L Input/Output Characteristics

COP410L/411L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410L/411L instruction set.

TABLE II. COP410L/411L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP410L/411L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP410L/411L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM (PC_8,A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	0110 | 000 | a_8	a → PC	None	Jump
		– –	$a_{7:0}$			
JP	a	– –	1 | $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 3)
			or			
		– –	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 | $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6–	0110 | 100 | a_8	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
		– –	$a_{7:0}$			
RET		48	0100 | 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 | 0011	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
		3C	0011 | 1100			
LD	r	–5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8,A,M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23	0010 | 0011	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
		BF	1011 | 1111			
XDS	r	–7	00 | r | 0111	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	00 | r | 0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r

Instruction Set (Continued)

TABLE III. COP410L/411L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	– –	00 | r | (d − 1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6–	0011 | 0011 0110 | y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 | 0011 0010 | 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0 = 0$	
	1	11	0001 | 0001		$G_1 = 0$	
	2	03	0000 | 0011		$G_2 = 0$	
	3	13	0001 | 0011		$G_3 = 0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1 = 0$	
	2	03	0000 | 0011		$RAM(B)_2 = 0$	
	3	13	0001 | 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 | 0011 0010 | 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 | 0011 0010 | 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 | 0011 0011 | 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP410L/411L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP410L/411L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or 7 will access data in the next group of 4 pages.

Option List

The COP410L/411L mask-programmable options are assigned numbers which correspond with the COP410L pins.

The following is a list of COP410L options. The LED Direct Drive option on the L Lines cannot be used if higher V_{CC} option is selected. When specifying a COP411L chip, Option 2 must be set to 3, Options 20, 21, and 22 to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Output (no option available for COP411L)
- = 0: Clock output to ceramic resonator
- = 1: Pin is RAM power supply (V_R) input
- = 3: No connection

Option 3: CKI Input
- = 0: Oscillator input divided by 8 (500 kHz max)
- = 1: Single-pin RC controlled oscillator divided by 4
- = 2: External Schmitt trigger level clock divided by 4

Option 4: $\overline{\text{RESET}}$ Input
- = 0: Load device to V_{CC}
- = 1: Hi-Z input

Option 5: L_7 Driver
- = 0: Standard output
- = 1: Open-drain output
- = 2: High current LED direct segment drive output
- = 3: High current TRI-STATE push-pull output
- = 4: Low-current LED direct segment drive output
- = 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: Operating voltage

	COP41XL	COP31XL
= 0:	+4.5V to +6.3V	+4.5V to +5.5V
= 1:	+4.5V to +9.5V	+4.5V to +7.5V

Option 10: L_3 Driver
same as Option 5

Option 11: L_2 Driver
same as Option 5

Option 12: L_1 Driver
same as Option 5

Option 13: L_0 Driver
same as Option 5

Option 14: SI Input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 15: SO Driver
- = 0: Standard Output
- = 1: Open-drain output
- = 2: Push-pull output

Option 16: SK Driver
same as Option 15

Option List (Continued)

Option 17: G_0 I/O Port
= 0: Standard output
= 1: Open-drain output

Option 18: G_1 I/O Port
same as Option 17

Option 19: G_2 I/O Port
same as Option 17

Option 20: G_3 I/O Port (no option available for COP411L)
same as Option 17

Option 21: D_3 Output (no option available for COP411L)
= 0: Very-high sink current standard output
= 1: Very-high sink current open-drain output
= 2: High sink current standard output
= 3: High sink current open-drain output
= 4: Standard LSTTL output (fanout = 1)
= 5: Open-drain LSTTL output (fanout = 1)

Option 22: D_2 Output (no option available for COP411L)
same as Option 21

Option 23: D_1 Output
same as Option 21

Option 24: D_0 Output
same as Option 21

Option 25: L Input Levels
= 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
= 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 26: G Input Levels
same as Option 25

Option 27: SI Input Levels
same as Option 25

Option 28: COP Bonding
= 0: COP410L (24-pin device)
= 1: COP411L (20-pin device)
= 2: Both 24- and 20-pin versions

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP410L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

Option Table

The following option information is to be sent to National along with the EPROM.

	Option Data	
OPTION 1 VALUE =	0	IS: GROUND PIN
OPTION 2 VALUE =		IS: CKO PIN
OPTION 3 VALUE =		IS: CKI INPUT
OPTION 4 VALUE =		IS: RESET INPUT
OPTION 5 VALUE =		IS: L(7) DRIVER
OPTION 6 VALUE =		IS: L(6) DRIVER
OPTION 7 VALUE =		IS: L(5) DRIVER
OPTION 8 VALUE =		IS: L(4) DRIVER
OPTION 9 VALUE =		IS: V_{CC} PIN
OPTION 10 VALUE =		IS: L(3) DRIVER
OPTION 11 VALUE =		IS: L(2) DRIVER
OPTION 12 VALUE =		IS: L(1) DRIVER
OPTION 13 VALUE =		IS: L(0) DRIVER
OPTION 14 VALUE =		IS: SI INPUT

	Option Data	
OPTION 15 VALUE =		IS: SO DRIVER
OPTION 16 VALUE =		IS: SK DRIVER
OPTION 17 VALUE =		IS: G_0 I/O PORT
OPTION 18 VALUE =		IS: G_1 I/O PORT
OPTION 19 VALUE =		IS: G_2 I/O PORT
OPTION 20 VALUE =		IS: G_3 I/O PORT
OPTION 21 VALUE =		IS: D_3 OUTPUT
OPTION 22 VALUE =		IS: D_2 OUTPUT
OPTION 23 VALUE =		IS: D_1 OUTPUT
OPTION 24 VALUE =		IS: D_0 OUTPUT
OPTION 25 VALUE =		IS: L INPUT LEVELS
OPTION 26 VALUE =		IS: G INPUT LEVELS
OPTION 27 VALUE =		IS: SI INPUT LEVELS
OPTION 28 VALUE =		IS: COPS BONDING

National Semiconductor

COP413L/COP313L Single Chip Microcontrollers

General Description

The COP413L and COP313L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These Control Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, 15 I/O lines with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Control Oriented Processor at a very low end-product cost.

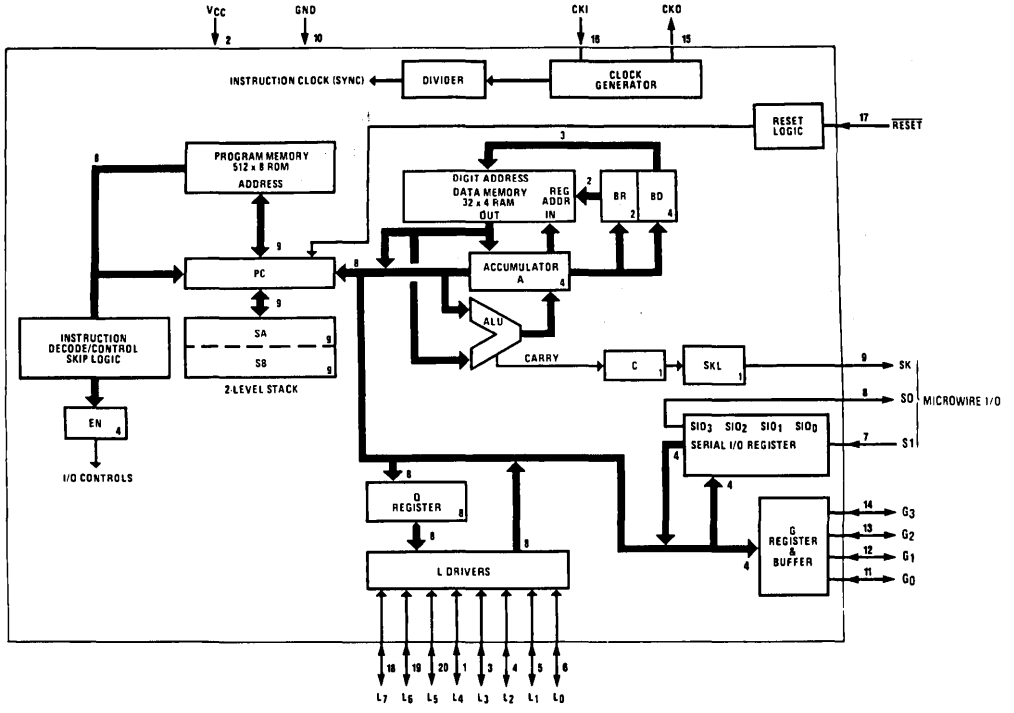
The COP313L is an exact functional equivalent but extended temperature version of the COP413L.

The COP401L-R13 and COP410L-X13 should be used for exact emulation.

Features

- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-Level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max.)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose outputs
- High noise immunity inputs (V_{IL} = 1.2V, V_{IH} = 3.6V)
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device COP313L (−40°C to +85°C)

Block Diagram



TL/DD/8371-1

FIGURE 1

COP413L Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.3 to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation COP413L	0.3 Watt at 70°C
Total Source Current	25 mA
Total Sink Current	25 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Power Supply Ripple	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})		3.0		V
Logic Low (V_{IL})			0.4	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low			0.6	V
SO Input Level (Test Mode)	(Note 2)	2.5		V
SI Input Level				
Logic High	(TTL Level)	2.0		V
Logic Low			0.8	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance			7	pF
Reset Input Leakage		−1	+1	μA
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{OL}=0.4V$	0.9		mA
L0–L7 Outputs, G0–G3	$V_{OL}=0.4V$	0.4		mA
CKO (I_{OL})	$V_{OL}=0.4V$	0.2		mA
Output Source Current				
L0–L7 and G0–G3	$V_{OH}=2.4V$	−25		μA
SO and SK Outputs (I_{OH})	$V_{OH}=1.0V$	−1.2		mA
Push-Pull	$V_{OH}=2.4V$	−25		μA
SI Input Load Source Current	$V_{IL}=0V$	−10	−140	μA
Total Sink Current Allowed				
L7–L4, G Port			4	mA
L3–L0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

COP313L Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.3 to +7V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation COP313L	0.20 Watt at 85°C
Total Source Current	25 mA
Total Sink Current	25 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})		3.0		V
Logic Low (V_{IL})			0.3	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low			0.4	V
SO Input (Test Mode)	(Note 2)	2.5		V
SI Input Level				
Logic High	(TTL Level)	2.2		V
Logic Low			0.6	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance			7	pF
Reset Input Leakage		−2	+2	μA
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{OL}=0.4V	0.8		mA
L0–L7 Outputs, G0–G3 (I_{OL})	V_{OL}=0.4V	0.4		mA
CKO (I_{OL})	V_{OL}=0.4V	0.2		mA
Output Source Current				
L0–L7 and G0–G3	V_{OH}=2.4V	−23		μA
SO and SK Outputs (I_{OH})	V_{OH}=1.0V	−1.0		mA
(Push-Pull)	V_{OH}=2.4V	−23		μA
SI Input Load Source Current	V_{IL}=0V	−10	−200	μA
Total Sink Current Allowed				
L7–L4, G Port			4	mA
L3–L0			4	mA
Any Other Pin			1.5	mA
Total Source Current Allowed Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

AC Electrical Characteristics

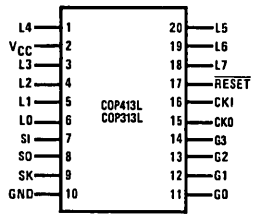
COP413L: $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 6.3V$
COP313L: $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time - t_c		16	40	μs
CKI				
Input Frequency - fi	÷8 Mode	0.2	0.5	MHz
Duty Cycle		30	60	%
Rise Time	fi = 0.5 MHz		500	ns
Fall Time			200	ns
CKI Using RC (÷4)	R = 56 kΩ ±5% C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
Inputs:				
G3–G0, L7–L0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3	1.3	μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: $C_L = 50$ pF, $R_L = 20$ kΩ, $V_{OUT} = 1.5V$			
SO, SK Outputs			4.0	μs
tpd1, tpd0				
All Other Outputs				
tpd1, tpd0			5.6	μs

Note 1: Variation due to the device included.

Connection Diagram

S.O. Wide and DIP



TL/DD/8371-2

FIGURE 2

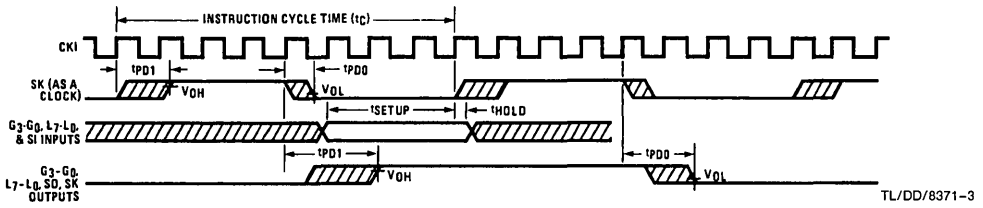
Order Number COP313L-XXX/D or COP413L-XXX/D
See NS Hermetic Package Number D20A

Order Number COP313L-XXX/WM or COP413L-XXX/WM
See NS Surface Mount Package Number M20B

Order Number COP313L-XXX/N or COP413L-XXX/N
See NS Molded Package Number N20A

Pin Descriptions

Pin	Description
L7–L0	8-bit bidirectional I/O port
G3–G0	4-bit bidirectional I/O port
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output or NC
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power Supply
GND	Ground



TL/DD/8371-3

FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)

Functional Description

A block diagram of the COP413L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP413L also apply to the COP313L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP413L instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction.

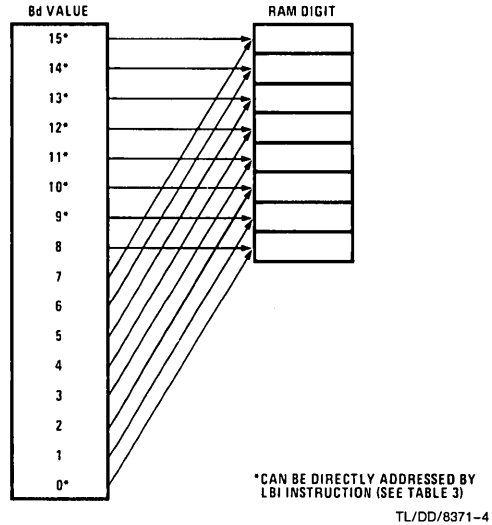
The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP413L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general purpose bidirectional I/O ports.



TL/DD/8371–4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0 selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting with each instruction cycle time. The data present at SO goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP413L operation.

Functional Description (Continued)

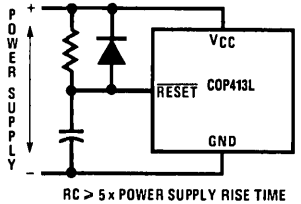
TABLE I. Enable Register Modes - Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1, SK=Clock If SKL=0, SK=0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL=1, SK=Clock If SKL=0, SK=0
0	1	Binary Counter	Input to Binary Counter	0	If SKL=1, SK=1 If SKL=0, SK=0
1	1	Binary Counter	Input to Binary Counter	1	If SKL=1, SK=1 If SKL=0, SK=0

3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{RESET}$ pin as shown below (*Figure 5*). The $\overline{RESET}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{RESET}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/8371-5

FIGURE 5. Power-Up Clear Circuit

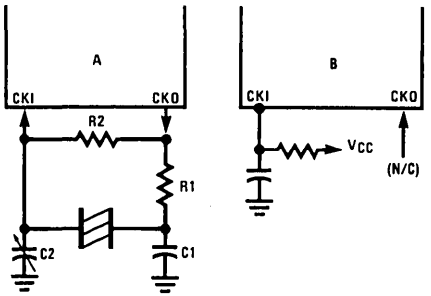
Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

OSCILLATOR

There are two basic clock oscillator configurations available as shown by *Figure 6*.

a. Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.

b. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO becomes no connection.



TL/DD/8371-6

FIGURE 6. COP413L Oscillator

Ceramic Resonator Oscillator

Resonator Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (in μs)
51	100	19 ± 15%
82	56	19 ± 13%

Note: 200 kΩ ≥ R ≥ 25 kΩ
220 pF ≥ C ≥ 50 pF

Functional Description (Continued)

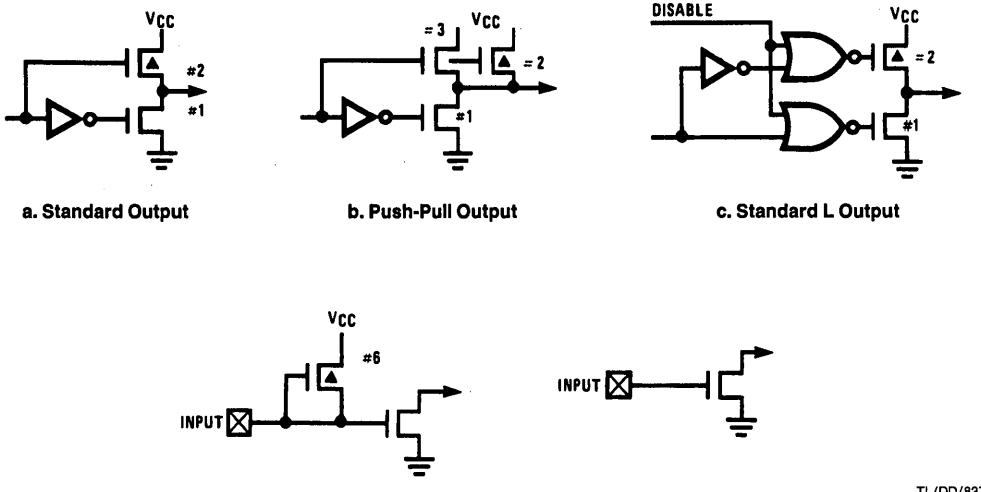


FIGURE 7. Input and Output Configurations

I/O CONFIGURATIONS

COP413L inputs and outputs have the following configurations, illustrated in *Figure 7*:

a. G0–G3—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}.

b. SO, SK—an enhancement mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

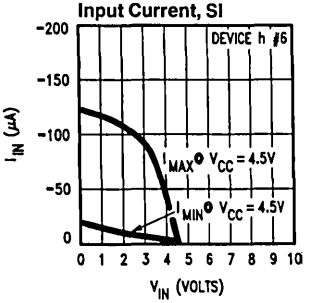
c. L0–L7—same as a., but may be disabled.

d. SI has on-chip depletion load device to V_{CC}.

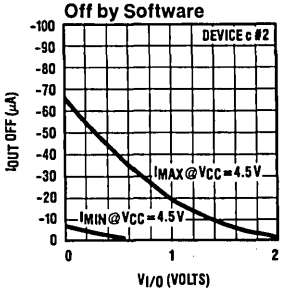
e. $\overline{\text{RESET}}$ has a Hi-Z input which must be driven to a "1" or "0" by external components.

Typical Performance Characteristics

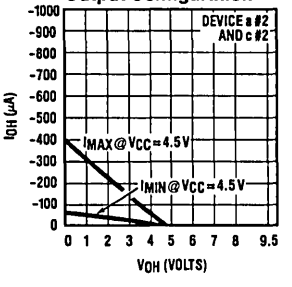
Input Current, SI



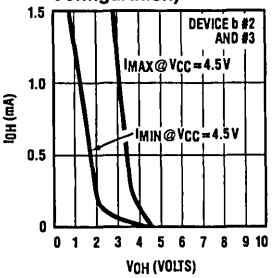
Input Current for L0 through L7 when Output Programmed Off by Software



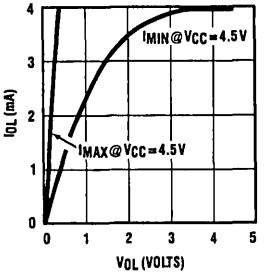
Source Current L7–L0, G3–G0 Standard Output Configuration



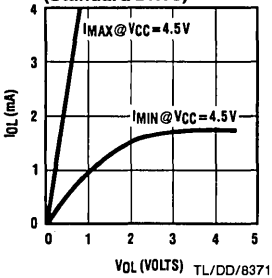
Source Current for SO and SK (Push-Pull Configuration)



Output Sink Current for SO and SK



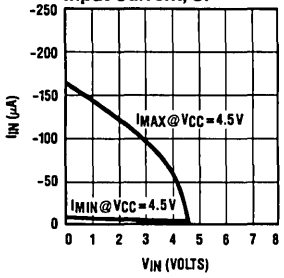
Output Sink Current for L0–L7, G0–G3 (Standard Drive)



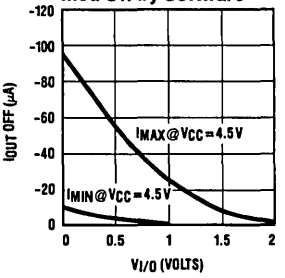
TL/DD/8371–8

FIGURE 8a. COP413L I/O DC Current Characteristics

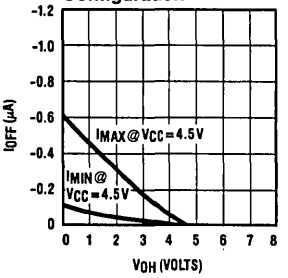
Input Current, SI



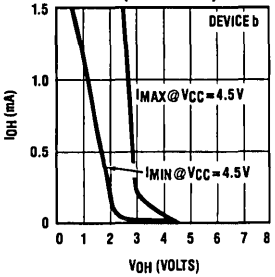
Input Current for L0–L7 when Output Programmed Off by Software



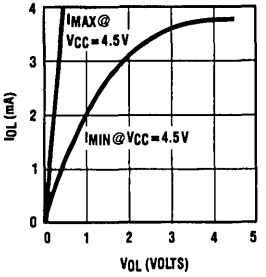
Source Current for L0–L7, G0–G3 Standard Output Configuration



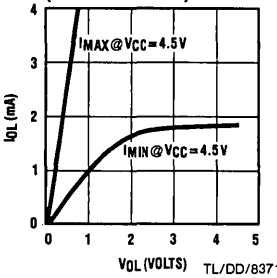
Source Current for SO and SK (Push-Pull)



Output Sink Current for SO and SK



Output Sink Current for L0–L7, G0–G3 (Standard Drive)



TL/DD/8371–9

FIGURE 8b. COP313L I/O DC Current Characteristics

COP413L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table. Table III provides the mnemonic, operand, machine code data flow, skip conditions and description associated with each instruction in the COP413L instruction set.

TABLE II. COP413L Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE® I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic Controlled Clock Output
Instruction Operand Symbols	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
Operational Symbols	
+	Plus
−	Minus
→	Replaces
⟷	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A+RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A+y → A	Carry	Add Immediate, Skip on Carry (y≠0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A⊕RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM(PC_8,A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	|0110|000|a_8|	a → PC	None	Jump
		–	|$a_{7:0}$|			
JP	a	–	|1|$a_{6:0}$| (pages 2, 3 only) or	a → $PC_{6:0}$	None	Jump within-Page (Note 3)
		–	|11|$a_{5:0}$| (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	|10|$a_{5:0}$|	PC+1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6–	|0110|100|a_8|	PC+1 → SA → SB a → PC	None	Jump to Subroutine
		–	|$a_{7:0}$|			
RET		48	|0100|1000|	SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	|0011|0011|	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	|0011|1100|	RAM(B) → $Q_{3:0}$		
LD	r	–5	|00|r|0101|	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	|1011|1111|	ROM(PC_8, A,M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	|0100|1100|	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	|0100|0101|	0 → $RAM(B)_1$		
	2	42	|0100|0010|	0 → $RAM(B)_2$		
	3	43	|0100|0011|	0 → $RAM(B)_3$		
SMB	0	4D	|0100|1101|	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	|0100|0111|	1 → $RAM(B)_1$		
	2	46	|0100|0110|	1 → $RAM(B)_2$		
	3	4B	|0100|1011|	1 → $RAM(B)_3$		

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7–	0111 | y	y → RAM(B) Bd+1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br⊕r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23	0010 | 0011	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
		BF	1011 | 1111			
XDS	r	–7	00 | r | 0111	RAM(B) ⟷ A Bd–1 → Bd Br⊕r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	–4	00 | r | 0100	RAM(B) ⟷ A Bd+1 → Bd Br⊕r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	–	00 | r | (d–1) (d=0,9:15)	r,d → B	Skip until not a LBI	Load B immediate with r,d (Note 5)
LEI	y	33	0011 | 0011	y → EN	None	Load EN Immediate (Note 6)
		6–	0110 | y			
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C="1"	Skip if C is True
SKE		21	0010 | 0001		A=RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 | 0011		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
		21	0010 | 0001			
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0=0$	
	1	11	0001 | 0001		$G_1=0$	
	2	03	0000 | 0011		$G_2=0$	
	3	13	0001 | 0011		$G_3=0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1=0$	
	2	03	0000 | 0011		$RAM(B)_2=0$	
	3	13	0001 | 0011		$RAM(B)_3=0$	

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 | 0011	G → A	None	Input G Ports to A
		2A	0010 | 1010			
INL		33	0011 | 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 | 1110	$L_{3:0}$ → A		
OMG		33	0011 | 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicity defined (e.g., Br and Bd are explicitly defined) Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1* e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP413L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM (B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP413L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or will access data in the next group of 4 pages.

Description of Selected Instructions (Continued)

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmable COP413L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

Option List

The option selected must be sent in with the EPROM of ROM Code for a Mask order of 413L. Make xerox copy of the table, select the appropriate option, and send it in with the EPROM.

COP 413L/COP 313L

Option 1: Oscillator Selection

=0 Ceramic Resonator or external input frequency divided by 8. CKO is oscillator output.

=1 Single pin RC controlled oscillator divided by 4. CKO is no connection.

NOTE:

The following option information is to be sent to National along with the EPROM

Option 1: Value = _____ is: Oscillator Selection

National Semiconductor

COP413C/COP413CH/COP313C/COP313CH Single-Chip CMOS Microcontrollers

General Description

The COP413C, COP413CH, COP313C, and COP313CH fully static, single-chip CMOS microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP413CH is identical to the COP413C except for operating voltage and frequency. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide a customized controller-oriented processor at a low end-product cost.

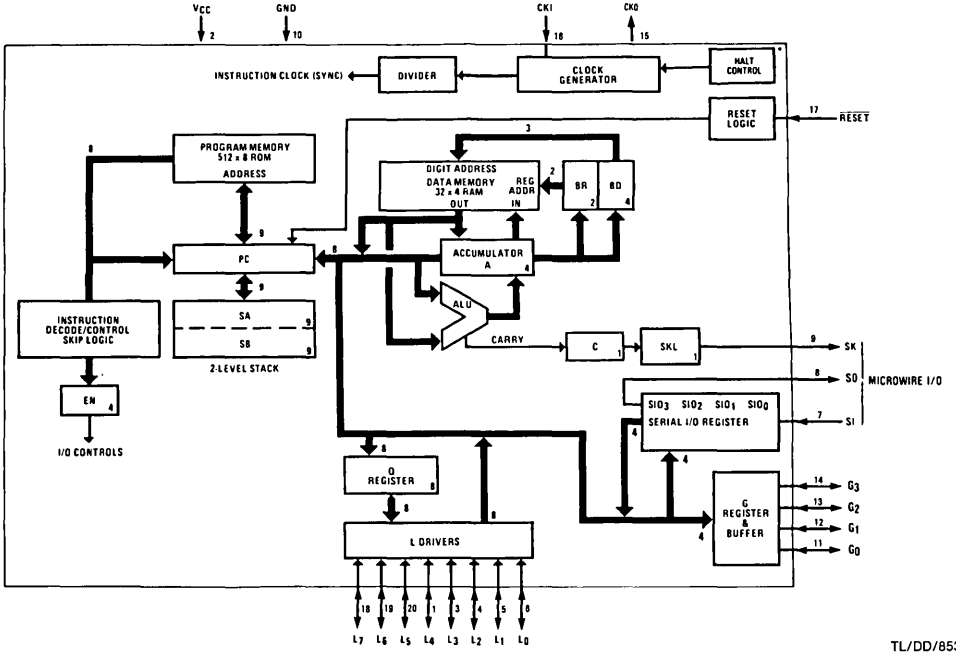
The COP313C/COP313CH is the extended temperature range version of the COP413C/COP413CH.

For emulation use the ROMless COP404C.

Features

- Lowest power dissipation (40 μW typical)
- Low cost
- Power-saving HALT Mode
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-level subroutine stack
- DC to 4 μs instruction time
- Single supply operation (3V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- Software/hardware compatible with other members of the COP400 family
- Extended temperature (−40°C to +85°C) devices available

Block Diagram



TL/DD/8537–1

FIGURE 1. COP413C/413CH

COP413C/COP413CH

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Conditions	COP413C		COP413CH		Units
		Min	Max	Min	Max	
Operating Voltage		3.0	5.5	4.5	5.5	V
Power Supply Ripple (Note 4)			0.1 V_{CC}		0.1 V_{CC}	V
Supply Current (Note 1)	V_{CC} = 5.0V, t_c = Min V_{CC} = 3.0V, t_c = Min (t_c is inst. cycle)		500 300		2000	µA µA
HALT Mode Current (Note 2)	V_{CC} = 5.0V, F_I = 0 kHz V_{CC} = 3.0V, F_I = 0 kHz		30 10		30	µA µA
Input Voltage Levels RESET, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		 0.9 V_{CC} 0.7 V_{CC}	 0.1 V_{CC} 0.2 V_{CC}	 0.9 V_{CC} 0.7 V_{CC}	 0.1 V_{CC} 0.2 V_{CC}	 V V V V
RESET, SI Input Leakage		−1	+1	−1	+1	µA
Input Capacitance			7		7	pF
Output Voltage Levels (SO, SK, L Port) Logic High Logic Low	 I_{OH} = −10 µA I_{OL} = 10 µA	 V_{CC} − 0.2	 0.2	 V_{CC} − 0.2	 0.2	 V V
Output Current Levels Sink (Note 3) Source (SO, SK, L Port) Source (G Port)	 V_{CC} = Min, V_{OUT} = V_{CC} V_{CC} = Min, V_{OUT} = 0V V_{CC} = Min, V_{OUT} = 0V	 0.2 −0.1 −8	 −150	 1.2 −0.5 −30	 −330	 mA mA µA
Allowable Sink/Source Current Per Pin (Note 3)			5		5	mA
TRI-STATE Leakage Current		−2	+2	−2	+2	µA

COP413C/COP413CH

AC Electrical Characteristics $0°C \leq T_A \leq 70°C$ unless otherwise specified

Parameter	Conditions	COP413C		COP413CH		Units
		Min	Max	Min	Max	
Instruction Cycle Time		16	DC	4	DC	μs
Operating CKI Frequency	÷8 Mode	DC	500	DC	2000	kHz
Instruction Cycle Time RC Oscillator ÷ 4	R = 30k ±5%, V_{CC} = 5V C = 82 pF ± 5%			8	16	μs
Instruction Cycle Time RC Oscillator ÷ 4 (Note 6)	R = 56k ±5%, V_{CC} = 5V C = 100 pF ± 5%	16	32	16	32	μs
Duty Cycle (Note 5)	Fi = Max freq ext clk	40	60	40	60	%
Rise Time (Note 5)	Fi = Max freq ext clk		60		60	ns
Fall Time (Note 5)	Fi = Max freq ext clk		40		40	ns
Inputs (See *Figure 3*)						
t_{SETUP}	G Inputs	tc/4 + 2.8		tc/4 + 0.7		μs
	SI Input	1.2		0.3		μs
	L Inputs	6.8		1.7		μs
t_{HOLD}		1.0		0.25		μs
Output Propagation Delay t_{PD1}, t_{PD0}	V_{OUT} = 1.5, C_L = 100 pF R_L = 5k		4.0		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating.

Note 3: SO output sink current must be limited to keep V_{OL} less tha 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 4: Voltage change must be less than 0.5V in a 1 ms period.

Note 5: This parameter is only sampled and not 100% tested.

Note 6: Variation due to the device included.

COP313C/COP313CH

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	COP313C		COP313CH		Units
		Min	Max	Min	Max	
Operating Voltage		3.0	5.5	4.5	5.5	V
Power Supply Ripple (Note 4)			0.1 V_{CC}		0.1 V_{CC}	V
Supply Current (Note 1)	V_{CC} = 5.0V, t_c = Min V_{CC} = 3.0V, t_c = Min (t_c is inst. cycle)		600 360		2500	μA μA
Halt Mode Current (Note 2)	V_{CC} = 5.0V, Fi = 0 kHz V_{CC} = 3.0V, Fi = 0 kHz		50 20		50	μA μA
Input Voltage Levels $\overline{RESET}$, CKI Logic High Logic Low All Other Inputs Logic High Logic Low		 0.9 V_{CC} 0.7 V_{CC}	 0.1 V_{CC} 0.2 V_{CC}	 0.9 V_{CC} 0.7 V_{CC}	 0.1 V_{CC} 0.2 V_{CC}	 V V V V
$\overline{RESET}$, SI Input Leakage		−2	+2	−2	+2	μA
Input Capacitance			7		7	pF
Output Voltage Levels (SO, SK, L Port) Logic High Logic Low	 I_{OH} = −10 μA I_{OL} = 10 μA	 V_{CC} − 0.2	 0.2	 V_{CC} − 0.2	 0.2	 V V
Output Current Levels Sink (Note 3) Source (SO, SK, L Port) Source (G Port)	 V_{CC} = Min, V_{OUT} = V_{CC} V_{CC} = Min, V_{OUT} = 0V V_{CC} = Min, V_{OUT} = 0V	 0.2 −0.1 −8	 −200	 1.2 −0.5 −30	 −440	 mA mA μA
Allowable Sink/Source Current Per Pin (Note 3)			5		5	mA
TRI-STATE Leakage Current[3]		−4	+4	−4	+4	μA

COP313C/COP313CH

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	COP313C		COP313CH		Units
		Min	Max	Min	Max	
Instruction Cycle Time		16	DC	4	DC	μs
Operating CKI Frequency	÷ 8 Mode	DC	500	DC	2000	kHz
Instruction Cycle Time RC Oscillator ÷ 4	R = 30k ±5%, V_{CC} = 5V C = 82 pF ± 5%			8	16	μs
Instruction Cycle Time RC Oscillator ÷ 4 (Note 6)	R = 56k ±5%, V_{CC} = 5V C = 100 pF ± 5%	16	32	16	32	μs
Duty Cycle (Note 5)	Fi = Max Freq Ext Clk	40	60	40	60	%
Rise Time (Note 5)	Fi = Max Freq Ext Clk		60		60	ns
Fall Time (Note 5)	Fi = Max Freq Ext Clk		40		40	ns
Inputs (See *Figure 3*)						
t_{SETUP}	G Inputs	tc/4 + 2.8		tc/4 + 0.7		μs
	SI Input	1.2		0.3		μs
	L Inputs	6.8		1.7		μs
t_{HOLD}		1.0		0.25		μs
Output Propagation Delay t_{PD1}, t_{PD0}	V_{OUT} = 1.5V, C_L = 100 pF R_L = 5k		4.0		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating.

Note 3: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

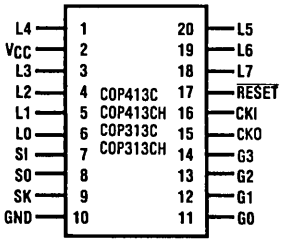
Note 4: Voltage change must be less than 0.5V in a 1 ms period.

Note 5: This parameter is only sampled and not 100% tested.

Note 6: Variation due to the device included.

Connection Diagram

DIP



TL/DD/8537-2

Top View

Pin Descriptions

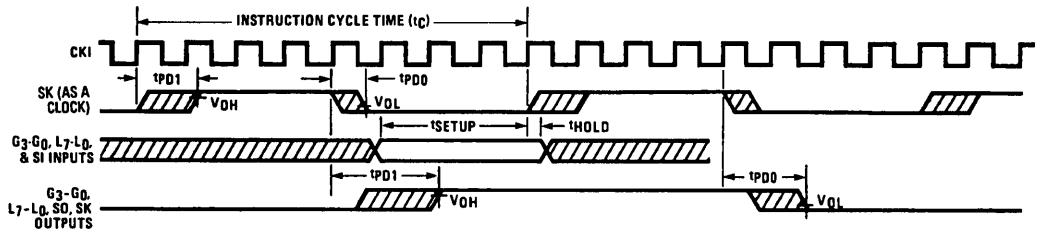
Pin	Description
L_7–L_0	8-bit bidirectional I/O port with TRI-STATE
G_3–G_0	4-bit bidirectional I/O port
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	Crystal oscillator output, or NC
$\overline{RESET}$	System reset input
V_{CC}	System power supply
GND	System Ground

FIGURE 2

Order Number COP313C-XXX/D, COP313CH-XXX/D, COP413C-XXX/D or COP413CH-XXX/D
See NS Hermetic Package Number D20A

Order Number COP313C-XXX/N, COP313CH-XXX/N, COP413C-XXX/N or COP413CH-XXX/N
See NS Molded Package Number N20A

Timing Waveform



TL/DD/8537–3

FIGURE 3. Input/Output Timing Diagrams (Divide-by-8 Mode)

Development Support

The MOLE (Microcontroller On Line Emulator) is a low cost development system and real time emulator for COPS' products. They also include TMP, 8050 and the new 16 bit HPC microcontroller family. The MOLE provides effective support for the development of both software and hardware in the user's application.

The purpose of the MOLE is to provide a tool to write and assemble code, emulate code for the target microcontroller and assist in debugging of the system.

The MOLE can be connected to various hosts, IBM PC, STARPLEX™, Kaypro, Apple and Intel systems, via RS-232 port. This link facilitates the up loading/down loading of code, supports host assembly and mass storage.

The MOLE consists of three parts; brain, personality and optional host software.

The brain board is the computing engine of the system. It is a self-contained computer with its own firmware which provides for all system operation, emulation control, communication, from programming and diagnostic operation. It has three serial ports which can be connected to a terminal, host system, printer, modem or to other MOLE's in a multi-MOLE environment.

The personality board contains the necessary hardware and firmware needed to emulate the target microcontroller. The emulation cable which replaces the target controller attaches to this board. The software contains a cross assembler and a communications program for up loading and down loading code from the MOLE.

MOLE Ordering Information

P/N	Description
MOLE-BRAIN	MOLE Computer Board
MOLE-COPS-PB1	COPS Personality Board
MOLE-XXX-YYY	Optional Software

Where XXX = COPS
YYY = Host System, IBM, Apple, KAY (Kaypro), CP/M

Functional Description

To ease reading of this description, only COP413C is referenced; however, all such references apply equally to COP413CH, COP313C, and COP313CH.

A block diagram of the COP413C is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP413C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8 × 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4.* The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP413C is designed to ensure fully static operation of the device.

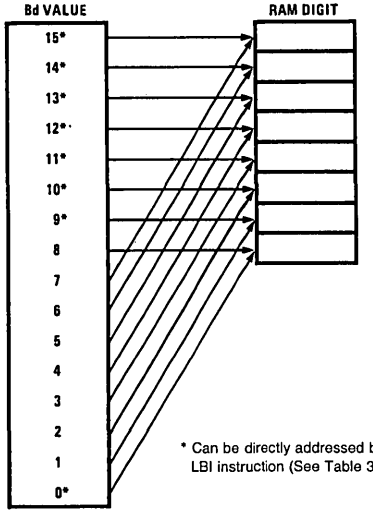
The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP413C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.



TL/DD/8537–4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP413C is MICROWIRE compatible.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

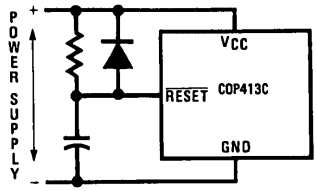
The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.
2. EN 1 is not used, it has no effect on the COP413C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

INITIALIZATION

The external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/8537–5

RC > 5 × Power Supply Rise Time
and RC > 100 × CKI Period

FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

HALT MODE

The COP413C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may be halted by the HALT instruction. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw more than a square-wave input. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP413C current drain.

$Ic = Iq + (V \times 20 \times Fi) + (V \times 1280 \times Fl/Dv)$

where Ic = chip current drain in microamps

Iq = quiescent leakage current (from curve)

Fl = CKI frequency in megahertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 8),

$Ic = 30 + (5 \times 20 \times 0.4) + (5 \times 1280 \times 0.4/8)$

$Ic = 30 + 40 + 320 = 390\ \mu A$

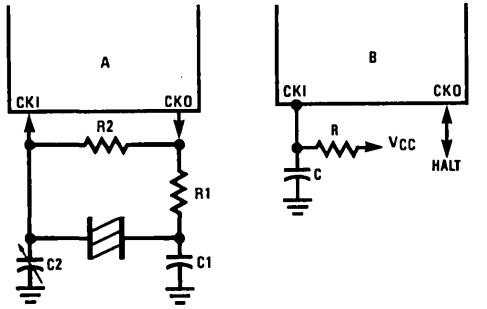
OSCILLATOR OPTIONS

There are two options available that define the use of CKI and CKO.

a. Cyrstal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 8.

b. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is NC.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.



TL/DD/8537-6

FIGURE 6. COP413C Oscillator

Crystal or Resonator					RC-Controlled Oscillator			
Crystal Value	Component Value R1	R2	C1 pF	C2 pF	R	C	Cycle Time	V_{CC}
32 kHz	220k	20M	30	5–36	15k	82 pF	4–9 μs	≥ 4.5V COP413CH Only
455 kHz	5k	10M	80	40	30k	82 pF	8–16 μs	≥ 4.5V COP413CH Only
2.000 MHz	2k	1M	30	6–36	47k	100 pF	16–32 μs	3.0 to 4.5V COP413C Only
					56k	100 pF	16–32 μs	≥ 4.5V
					Note: 15k ≤ R ≤ 150k, 50 pF ≤ C ≤ 150 pF			

Functional Description (Continued)

I/O CONFIGURATIONS

COP413C outputs have the following configurations, illustrated in *Figure 7*:

a. Standard SO, SK Output. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

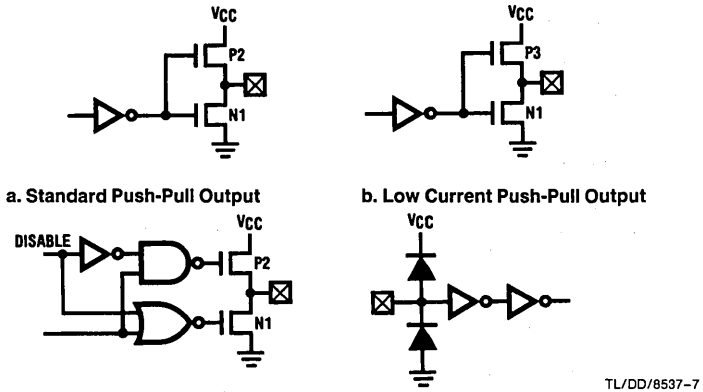
b. Low Current G Output. This is the same configuration as (a) above except that the sourcing current is much less.

c. Standard TRI-STATE L Output. L output is a CMOS output buffer similar to (a) which may be disabled by program control.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (*Figure 7d*).

When using the G I/O port as an input, set the output register to a logic "1" level. The P-channel device will act as a pull-up load. When using the L I/O port as an input, disable the L drivers with the LEI instruction. The drivers are then in TRI-STATE mode and can be driven externally.

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.



a. Standard Push-Pull Output

b. Low Current Push-Pull Output

c. Standard TRI-STATE "L" Output

d. Hi-Z Input

FIGURE 7. I/O Configurations

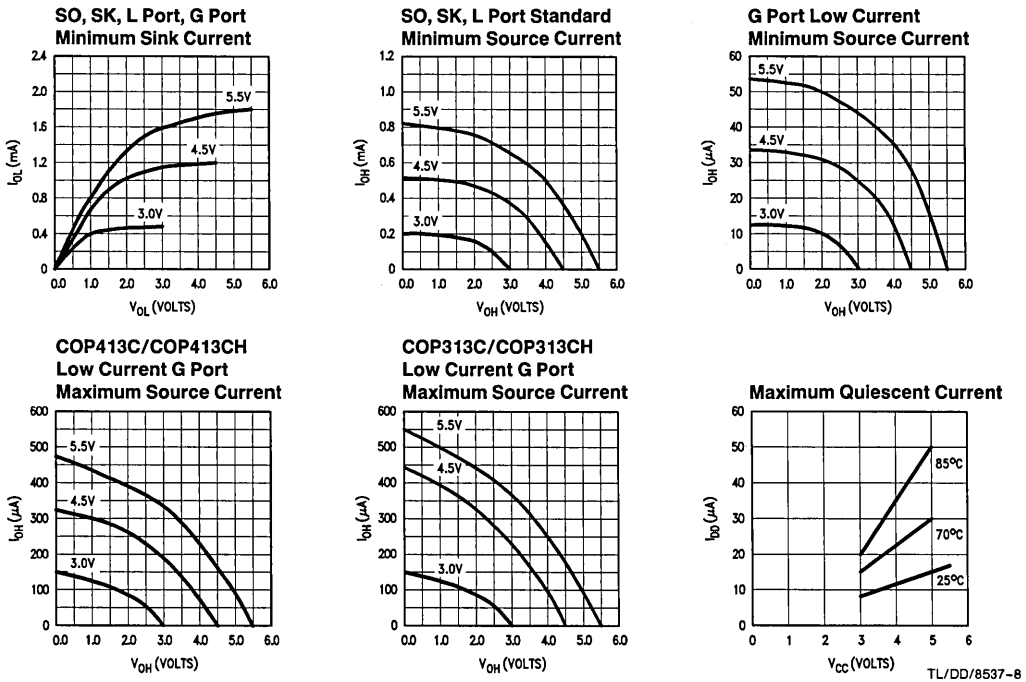


FIGURE 8

COP413C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP413C instruction set.

TABLE II. COP413C Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP413C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP413C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	0110 | 000 | a_8	a → PC	None	Jump
		–	$a_{7:0}$			
JP	a	–	1 | $a_{6:0}$ (pages 2, 3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 1)
			or			
		–	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	10 | $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 2)
JSR	a	6–	0110 | 100 | a_8	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
		–	$a_{7:0}$			
RET		48	0100 | 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 10011	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33	0011 | 0011		None	Halt processor
		38	0011 | 1000			
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 | 0011	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
		3C	0011 | 1100			
CQMA		33	0011 | 0011	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
		2C	0010 | 1100			
LD	r	–5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8,A,M) → Q SA → SB	None	Load Q Indirect
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23	0010 | 0011	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
		BF	1011 | 1111			

Instruction Set (Continued)

TABLE III. COP413C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	−7	|00|r|0111|	RAM(B) ⟷ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	−4	|00|r|0100|	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	–	|00|r|(d - 1)| (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6–	|0011|0011| |0010| y |	y → EN	None	Load EN Immediate
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	$G_0 = 0$	
	1	11	|0001|0001|	2nd byte	$G_1 = 0$	
	2	03	|0000|0011|	2nd byte	$G_2 = 0$	
	3	13	|0010|0011|	2nd byte	$G_3 = 0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1 = 0$	
	2	03	|0000|0011|		$RAM(B)_2 = 0$	
	3	13	|0001|0011|		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	|0011|0011| |0010|1010|	G → A	None	Input G Ports to A
INL		33 2E	|0011|0011| |0010|1110|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OMG		33 3A	|0011|0011| |0011|1010|	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	|0100|1111|	A ⟷ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP413C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

INSTRUCTION SET NOTES

a. The first word of a COP413C program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).

c. The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

COPS Programming Manual

For detailed information on writing. COPS programs, the COPS Programming Manual 424410284-001 provides an in-depth discussion of the COPS architecture, instruction set and general techniques of COPS programming. This manual is written with the programmer in mind.

OPTION LIST—OSCILLATOR SELECTION

The oscillator option selected must be sent in with the EPROM of ROM Code for masking into the COP413C. Select the appropriate option, make a photocopy of the table and send it with the EPROM.

COP413C/COP313C

Option 1: Oscillator selection

= 0 Ceramic Resonator input frequency divided by 8. CKO is oscillator output.

= 1 Single pin RC controlled oscillator divided by 4. CKO is no connection.

Note: The following option information is to be sent to National along with the EPROM.

Option 1: Value = ____ is Oscillator Selected.

National Semiconductor

PRELIMINARY

COP414L/COP314L Single-Chip N-Channel Microcontrollers

General Description

The COP414L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. This Controller Oriented Processor is a complete microcomputer containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP414L is an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

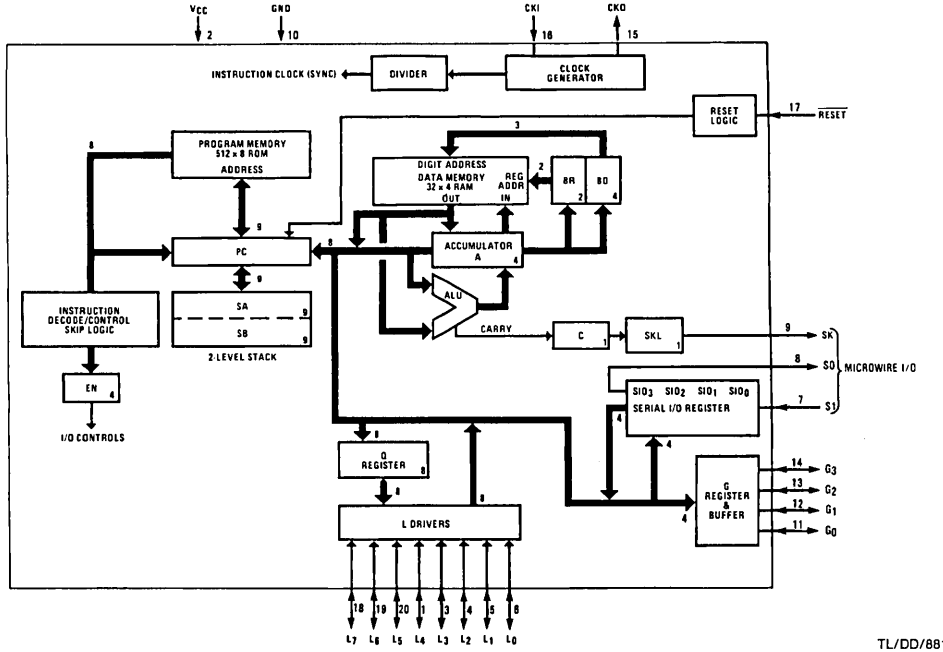
The COP314L is an exact functional equivalent but extended temperature version of COP414L.

The COP414L can be emulated by the COP404C. The COP401L should be used for exact emulation.

Features

- Late waferfab programming of ROM and I/O for fast delivery of units
- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device — COP314L (−40°C to +85°C)
- Wider supply range (4.5V–9.5V) optionally available

Block Diagram



TL/DD/8814–1

FIGURE 1. COP414L

COP414L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	
COP414L	0.65W at 25°C 0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Optional Operating Voltage (V_{CC})		4.5	9.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

COP414L

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Ouputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	1.8		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	1.2		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.9		mA
L_0–L_7 Outputs, G_0–G_3 and	$V_{CC} = 9.5V, V_{OL} = 0.4V$	0.4		mA
LSTTL D_0–D_3 Outputs (I_{OL})	$V_{CC} = 6.3V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
CKI (Single-pin RC Oscillator)	$V_{CC} = 4.5, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−140	−800	μA
All Outputs (I_{OH})	$V_{CC} = 6.3V, V_{OH} = 2.0V$	−75	−480	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−30	−250	μA
Push-Pull Configuration	$V_{CC} = 9.5V, V_{OH} = 4.75V$	−1.4		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 6.3V, V_{OH} = 2.4V$	−1.4		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−140	μA
Open Drain Output Leakage		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pin			1.5	mA

COP314L

Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation COP314L	0.65W at 25°C 0.20W at 85°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

COP314L: $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Optional Operating Voltage (V_{CC})		4.5	7.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Crystal Input				
Logic High (V_{IH})	V_{CC} = Max	3.0		
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

COP314L

DC Electrical Characteristics (Continued)

COP314L: $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs(I_{OL})	$V_{CC} = 7.5V, V_{OL} = 0.4V$	1.4		mA
	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0–L_7 Outputs, G_0–G_3 and	$V_{CC} = 7.5V, V_{OL} = 0.4V$	0.4		mA
LSTTL, D_0–D_3 Outputs (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
CKI (Single-pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	1.5		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−100	−900	μA
All Outputs (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	μA
Push-Pull Configuration	$V_{CC} = 7.5V, V_{OH} = 3.75V$	−0.85		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−200	μA
Open Drain Output Leakage		−5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pins			1.5	mA

AC Electrical Characteristics

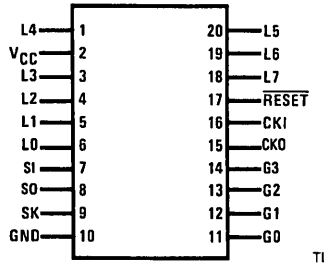
COP414L: 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 9.5V unless otherwise noted
COP314L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted
COP214L: −40°C ≤ T_A ≤ +110°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time — t_C		16	40	μs
CKI				
Input Frequency — f_I	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 0.5 MHz		500	ns
Fall Time			200	ns
CKI Using RC (÷4)	R = 56 kΩ ±5% C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
Inputs				
G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

Note 1: Variation due to the device included.

Connection Diagram

Dual-In-Line Package



TL/DD/8814–2

Top View

FIGURE 2

Order Number COP214L-XXX/D, COP314L-XXX/D or COP414L-XXX/D
See NS Hermetic Package D20A

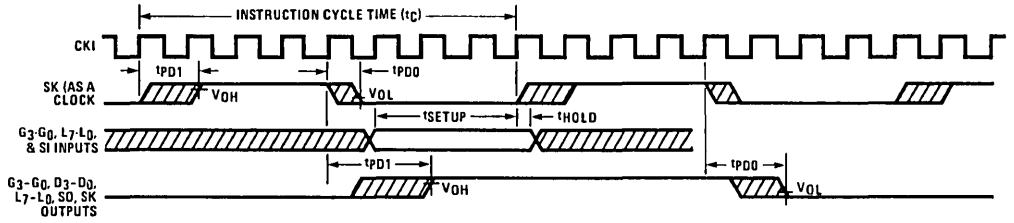
Order Number COP214L-XXX/N, COP314L-XXX/N or COP414L-XXX/N
See NS Molded Package N20A

Order Number COP214L-XXX/WM, COP314L-XXX/WM or COP414L-XXX/WM
See NS Surface Mount Package M20B

Pin Descriptions

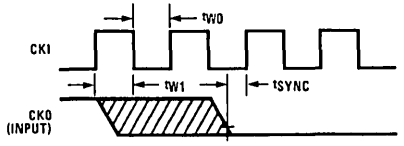
Pin	Description	Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE	CKI	System oscillator input
G_3–G_0	4 bidirectional I/O ports	CKO	System oscillator output
SI	Serial input (or counter input)	$\overline{\text{RESET}}$	System reset input
SO	Serial output (or general purpose output)	V_{CC}	Power supply
SK	Logic-controlled clock (or general purpose output)	GND	Ground

Timing Diagrams



TL/DD/8814-3

FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)



TL/DD/8814-4

FIGURE 3a. Synchronization Timing

Functional Description

A block diagram of the COP414L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP414L also apply to the COP314L, and COP214L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP414L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

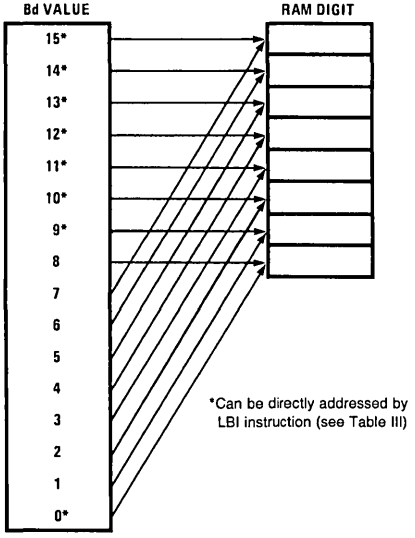
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).



TL/DD/8814-5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP414L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

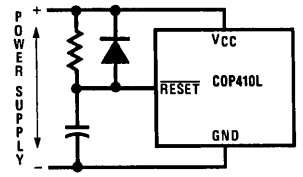
The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occuring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP414L operation.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below *(Figure 5)*. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



RC ≥ 5 × Power Supply Rise Time TL/DD/8814–6

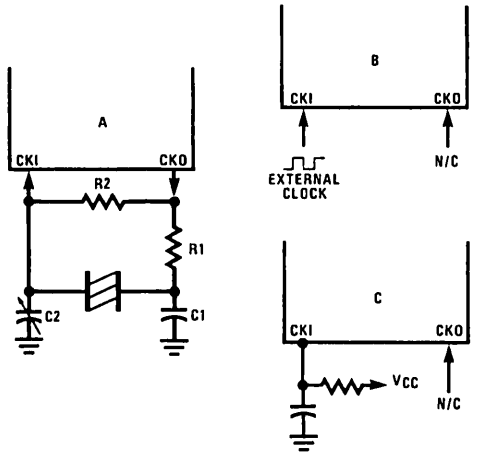
FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/8814–7

Ceramic Resonator Oscillator

Resonator Value	Components Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time in μs
51	100	19 ±15%
82	56	19 ±13%

Note: 200 kΩ ≥ R ≥ 25 kΩ. 360 pF ≥ C ≥ 50 pF. Does not include tolerances.

FIGURE 6. COP414L Oscillator

OSCILLATOR

There are four basic clock oscillator configurations available as shown by *Figure 6.*

a. Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.

b. External Oscillator. CKI is an external clock input signal. The external frequency is divided by 4 to give the instruction frequency time. CKO is no connection.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is no connection.

CKO PIN OPTIONS

In a resonator controlled oscillator system, CKO is used as an output to the resonator network. CKO is no connection for External or RC controlled oscillator.

I/O OPTIONS

COP414L inputs and outputs have the following optional configurations, illustrated in *Figure 7*:

a. Standard—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK and all D and G outputs.

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK and all D and G outputs.

c. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. Standard L—same as **a.**, but may be disabled. Available on L outputs only.

e. Open Drain L—same as **b.**, but may be disabled. Available on L outputs only.

f. An on-chip depletion load device to V_{CC}.

g. A Hi-Z input which must be driven to a "1" or "0" by external components.

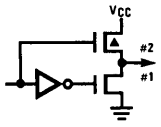
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP414L system.

The SO, SK outputs can be configured as shown in **a., b.,** or **c.** The G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in **d.,** or **e.**

An important point to remember if using configuration **d.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 8*, device 2.) However, when the L port is used as input, the disabled depletion device CANNOT be relied on to source sufficient current to pull an input to a logic "1".

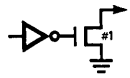
Functional Description (Continued)

a. Standard Output



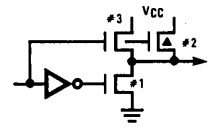
TL/DD/8814-8

b. Open-Drain Output



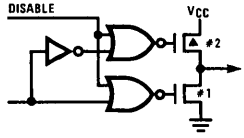
TL/DD/8814-9

c. Push-Pull Output



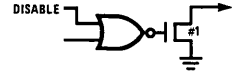
TL/DD/8814-10

d. Standard L Output



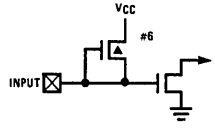
TL/DD/8814-11

e. Open-Drain L Output



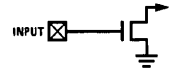
TL/DD/8814-12

f. Input with Load



TL/DD/8814-13

g. Hi-Z Input

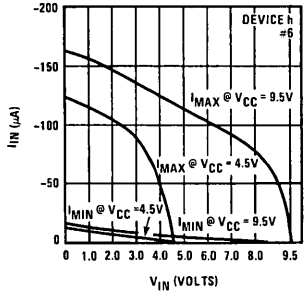


TL/DD/8814-14

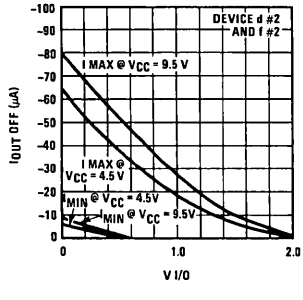
FIGURE 7. Input and Output Configurations

Typical Performance Curves

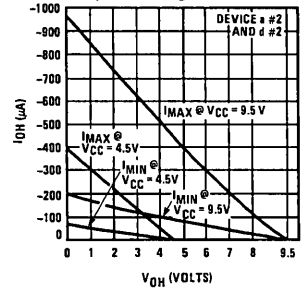
Input Current $\overline{\text{RESET}}$, SI



Input Current for L0 through L7 when Output Programmed Off by Software

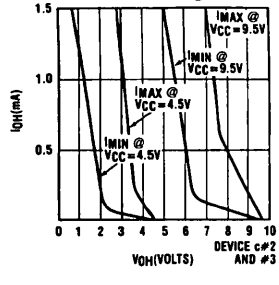


Source Current for Standard Output Configuration

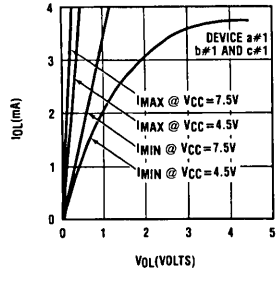


TL/DD/8814-15

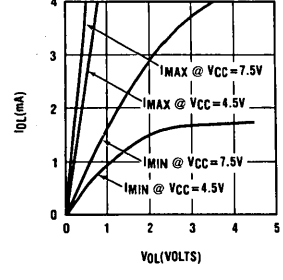
Source Current for SO and SK in Push-Pull Configuration



Output Sink Current For SO and SK



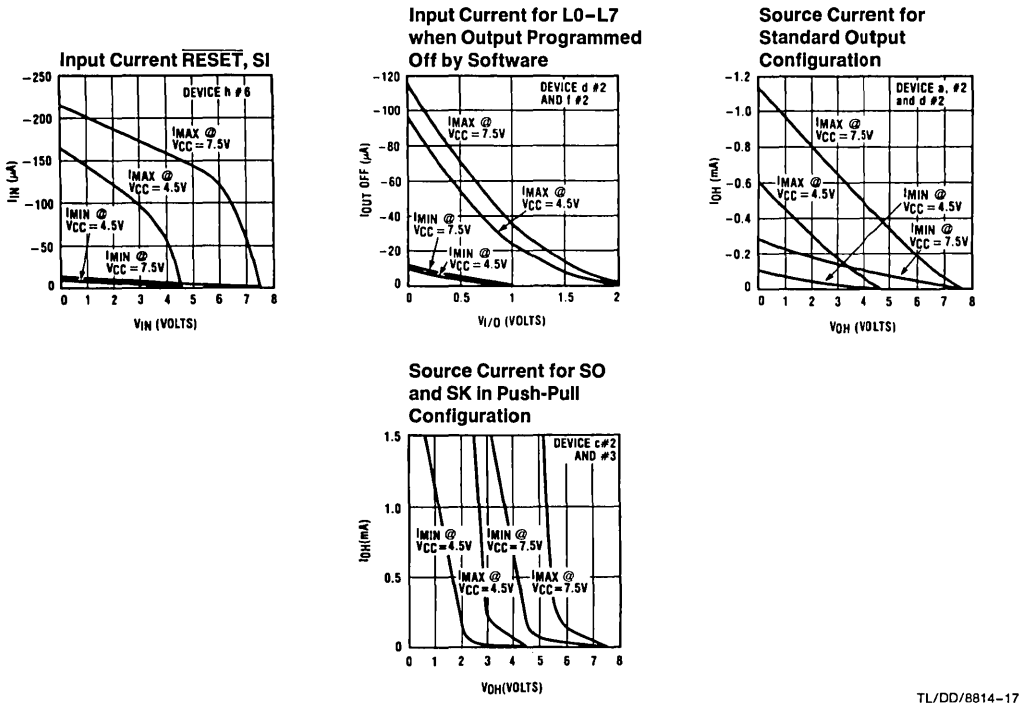
Output Sink Current for L0–L7, and Standard Drive Option for G0–G3



TL/DD/8814-16

FIGURE 8a. COP414 I/O DC Current Characteristics

Typical Performance Curves (Continued)



TL/DD/8814–17

FIGURE 8b. COP314L Input/Output Characteristics

COP414L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP414L instruction set.

TABLE II. COP414L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP414L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

COP414L Instruction Set (Continued)

TABLE III. COP414L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM (PC_8,A,M) $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– – –	0110 | 000 | a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	– –	1 | $a_{6:0}$ (pages 2, 3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 3)
		– –	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 | $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– – –	0110 | 100 | a_8 $a_{7:0}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100 | 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 | 0011 0011 | 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
LD	r	–5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8, A, M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0 1 2 3	4C 45 42 43	0100 | 1100 0100 | 0101 0100 | 0010 0100 | 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 | 1101 0100 | 0111 0100 | 0110 0100 | 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3, 15	23 BF	0010 | 0011 1011 | 1111	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
XDS	r	–7	00 | r | 0111	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	00 | r | 0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r

COP414L Instruction Set (Continued)

TABLE III. COP414L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	– –	00 | r | (d − 1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6–	0011 | 0011 0010 | y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 | 0011 0010 | 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0 = 0$	
	1	11	0001 | 0001		$G_1 = 0$	
	2	03	0000 | 0011		$G_2 = 0$	
	3	13	0001 | 0011		$G_3 = 0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1 = 0$	
	2	03	0000 | 0011		$RAM(B)_2 = 0$	
	3	13	0001 | 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 | 0011 0010 | 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 | 0011 0010 | 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 | 0011 0011 | 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Option List

The COP414L mask-programmable options are assigned numbers which correspond with the COP414L pins.

The following is a list of COP414L options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: L_4 Driver
- = 0: Standard output
- = 1: Open-drain output

Option 2: V_{CC} Pin
- = 0: Standard V_{CC}
- = 1: Optional higher voltage V_{CC}

Option 3: L_3 Driver
same as Option 1

Option 4: L_2 Driver
same as Option 1

Option 5: L_1 Driver
same as Option 1

Option 6: L_6 Driver
same as Option 1

Option 7: SI Input
- = 0: load device to V_{CC}
- = 1: Hi-Z Output

Option 8: SO Driver
- = 0: Standard output
- = 1: Open-drain output
- = 2: Push-pull output

Option 9: SK Driver
same as Option 8

Option 10:
- = 0: Ground Pin—no options available

Option 11: G_0 I/O Port
- = 0: Standard output
- = 1: Open-drain output

Option 12: G_1 I/O Port
same as Option 11

Option 13: G_2 I/O Port
same as Option 11

Option 14: G_3 I/O Port
same as Option 11

Option 15: CKO Output
- = 0: Clock output to ceramic resonator/crystal
- = 1: No connection

Option 16: CKI Input
- = 0: Ocillator input divided by 8 (500 kHz max)
- = 1: Single pin RC controlled oscillator divided by 4
- = 2: External Schmitt trigger level clock divided by 4

Option 17: $\overline{\text{RESET}}$ Input
- = 0: Load device to V_{CC}
- = 1: Hi-Z Input

Option 18: L_7 Driver
same as Option 1

Option 19: L_6 Driver
same as Option 1

Option 20: L_6 Driver
same as Option 1

Option 21: L Input Levels
- = 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
- = 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 22: G Input Levels
same as Option 21

Option 23: SI Input Levels
same as Option 21

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP414L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing tests only.

COP414L Option List

Please fill out the Option List and send it with the EPROM.

Option Data

OPTION 1 VALUE = _____ IS: L_4 DRIVER
OPTION 2 VALUE = _____ IS: V_{CC} PIN
OPTION 3 VALUE = _____ IS: L_3 DRIVER
OPTION 4 VALUE = _____ IS: L_2 DRIVER
OPTION 5 VALUE = _____ IS: L_1 DRIVER
OPTION 6 VALUE = _____ IS: L_6 DRIVER
OPTION 7 VALUE = _____ IS: SI INPUT
OPTION 8 VALUE = _____ IS: SO DRIVER
OPTION 9 VALUE = _____ IS: SK DRIVER
OPTION 10 VALUE = 0 IS: GROUND PIN
OPTION 11 VALUE = _____ IS: G_0 I/O PORT
OPTION 12 VALUE = _____ IS: G_1 I/O PORT
OPTION 13 VALUE = _____ IS: G_2 I/O PORT
OPTION 14 VALUE = _____ IS: G_3 I/O PORT
OPTION 15 VALUE = _____ IS: CKO OUTPUT
OPTION 16 VALUE = _____ IS: CKI INPUT
OPTION 17 VALUE = _____ IS: RESET INPUT
OPTION 18 VALUE = _____ IS: L_7 DRIVER
OPTION 19 VALUE = _____ IS: L_6 DRIVER
OPTION 20 VALUE = _____ IS: L_6 DRIVER
OPTION 21 VALUE = ———— IS: L INPUT LEVELS
OPTION 22 VALUE = _____ IS: G INPUT LEVELS
OPTION 23 VALUE = _____ IS: SI INPUT LEVELS

National Semiconductor

COP420/COP421/COP422 and COP320/COP321/COP322 Single-Chip N-Channel Microcontrollers

General Description

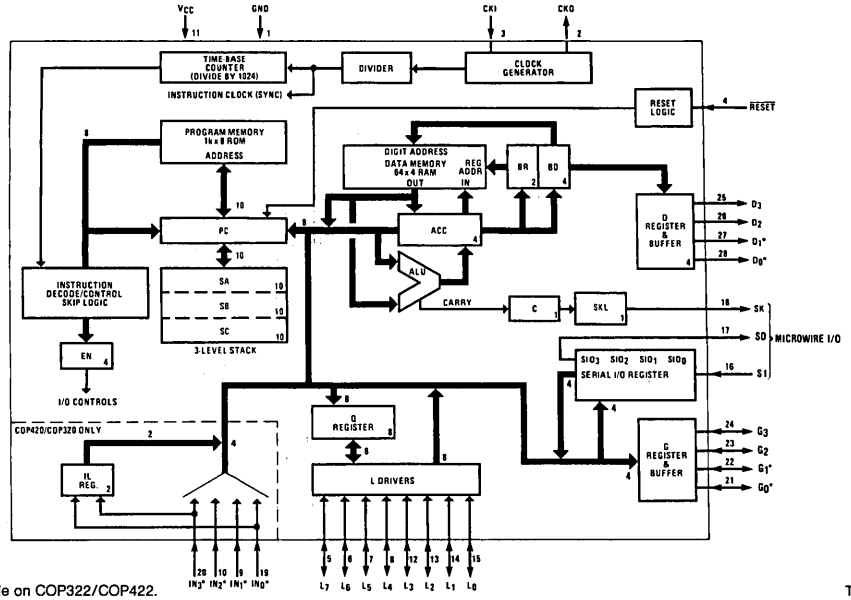
The COP420, COP421, COP422, COP320, COP321 and COP322 Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. They are complete microcomputers containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP421 is identical to the COP420, except with 19 I/O lines instead of 23; the COP422 has 15 I/O lines. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

The COP320 is the extended temperature range version of the COP420 (likewise the COP321 and COP322 are the extended temperature range versions of the COP421/COP422). The COP320/321/322 are exact functional equivalents of the COP420/421/422.

Features

- Low cost
- Powerful instruction set
- 1k x 8 ROM, 64 x 4 RAM
- 23 I/O lines (COP420, COP320)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 4.0 μs instruction time
- Single supply operation
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O capacity
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- LED direct drive outputs
- MICROBUS™ compatible
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device COP320/COP321/COP322 (−40°C to +85°C)

Block Diagram



*Not available on COP322/COP422.

TL/DD/6921-1

FIGURE 1

COP420/COP421/COP422 and COP320/COP321/COP322 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin	−0.3V to +7V
Operating Temperature Range	
COP420/COP421/COP422	0°C to 70°C
COP320/COP321/COP322	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Total Sink Current	75 mA
Total Source Current	95 mA
Package Power Dissipation	750 mW at 25°C
24 and 28 pin	400 mW at 70°C
	250 mW at 85°C
Package Power Dissipation	650 mW at 25°C
20 pin	300 mW at 70°C
	200 mW at 85°C
Lead Temperature (soldering, 10 sec.)	300°C

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP420/COP421/COP422

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	6.3	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	Outputs Open		38	mA
Supply Current	Outputs Open, $V_{CC} = 5V$, $T_A = 25°C$		30	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High	V_{CC} = Max.	3.0		
Logic High	$V_{CC} = 5V \pm 5\%$	2.0		V
Logic Low		−0.3	0.4	V
TTL Input	$V_{CC} = 5V \pm 5\%$			
Logic High		2.0		V
Logic Low		−0.3	0.8	V
Schmitt Trigger Inputs RESET, CKI (÷4)				
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	3.0	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	$V_{CC} = 5V \pm 5\%$	2.0		V
Logic Low		−0.3	0.8	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Load Source Current	$V_{CC} = 5V$, $V_{IN} = 0V$			
CKO		−4	−800	μA
All Others		−100	−800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
Standard Outputs				
TTL Operation	$V_{CC} = 5V \pm 10\%$			
Logic High	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low	$I_{OL} = 1.6$ mA	−0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	$I_{OH} = -10\ \mu A$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10\ \mu A$		0.2	V

Note 1: TRI-STATE and LED configurations are excluded.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

COP420/COP421/COP422

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
LED Direct Drive Output	$V_{CC} = 6V$			
Logic High	$V_{OH} = 2.0V$	2.5	14	mA
CKI Sink Current (R/C Option)	$V_{IN} = 3.5V$	2		mA
CKO (RAM Supply Current)	$V_R = 3.3V$		3	mA
TRI-STATE or Open Drain Leakage Current	$V_{CC} = 5V$	−2.5	+2.5	μA
Output Current Levels				
Output Sink Current (I_{OL})	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	+1.6		mA
Output Source Current (I_{OH})				
Standard Configuration				
All Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.0V$	−200	−900	μA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−100	−500	μA
Push-Pull Configuration				
SO, SK Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.0V$	−1.0		mA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−0.4		mA
TRI-STATE Configuration				
L_0–L_7 Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.2V$	−0.8		mA
	$V_{CC} = 4.5V$, $V_{OH} = 1.5V$	−0.9		mA
LED Configuration				
L_0–L_7 Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.0V$	−1.0		mA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−0.5		mA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

COP320/COP321/COP322

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	5.5	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	$T_A = -40°C$, Outputs Open		40	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High		2.2		V
Logic Low		−0.3	0.3	V
TTL Input	$V_{CC} = 5V \pm 5\%$			
Logic High		2.2		V
Logic Low		−0.3	0.6	V
Schmitt Trigger Inputs				
RESET, CKI (÷4)				
Logic High		$0.7\ V_{CC}$		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.0	3.0	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	$V_{CC} = 5V \pm 5\%$	2.2		V
Logic Low		−0.3	0.6	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Load Source Current	$V_{CC} = 5V$, $V_{IN} = 0V$			
CKO		−4	−800	μA
All Others		−100	−800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
Standard Outputs				
TTL Operation	$V_{CC} = 5V \pm 10\%$			
Logic High	$I_{OH} = -75\ \mu A$	2.4		V
Logic Low	$I_{OL} = 1.6$ mA	−0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	$I_{OH} = -10\ \mu A$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10\ \mu A$	−0.3	0.2	V
Output Current Levels				
LED Direct Drive Output	$V_{CC} = 5V$ (Note 4)			
Logic High	$V_{OH} = 2.0V$	1.0	12	mA
CKI Sink Current (R/C Option)	$V_{IN} = 3.5V$	2		mA
CKO (RAM Supply Current)	$V_R = 3.3V$		4	mA
TRI-STATE or Open Drain Leakage Current		−5	+5	μA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

Note 1: TRI-STATE and LED configurations are excluded.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

AC Electrical Characteristics

COP420/COP421/COP422 $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

COP320/COP321/COP322 $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	÷16 mode	1.6	4.0	MHz
	÷8 mode	0.8	2.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time	Freq. = 4 MHz		60	ns
Fall Time	Freq. = 4 MHz		40	ns
CKI Using RC *(Figure 8c)*	÷4 mode			
Frequency	R = 15 kΩ ±5%, C = 100 pF	0.5	1.0	MHz
Instruction Cycle Time (Note 5)		4	8	μs
CKO as SYNC Input *(Figure 8d)*				
t_{SYNC}	*Figure 3a*	50		ns
Inputs:				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
Output Propagation Delay	Test Conditions: R_L = 5 kΩ, C_L = 50 pF, V_{OUT} = 1.5V	300		ns
SO and SK				
t_{pd1}			1.0	μs
t_{pd0}			1.0	μs
CKO				
t_{pd1}			0.25	μs
t_{pd0}			0.25	μs
All Other Outputs				
t_{pd1}			1.4	μs
t_{pd0}			1.4	μs
MICROBUS™ Timing	C_L = 100 pF, V_{CC} = 5V ±5%			
Read Operation *(Figure 4)*				
Chip Select Stable before $\overline{RD}$—t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$—t_{RCS}		20		ns
$\overline{RD}$ Pulse Width—t_{RR}		400		ns
Data Delay from $\overline{RD}$—t_{RD}			375	ns
$\overline{RD}$ to Data Floating—t_{DF}			250	ns
Write Operation *(Figure 5)*				
Chip Select Stable before $\overline{WR}$—t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$—t_{WCS}		20		ns
$\overline{WR}$ Pulse Width—t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$—t_{DW}		320		ns
Data Hold Time for $\overline{WR}$—t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$—t_{WI}			700	ns

Note 1: Duty cycle = $t_{W1}/(t_{W1} + t_{W0})$.

Note 2: See *Figure 9* for additional I/O characteristics.

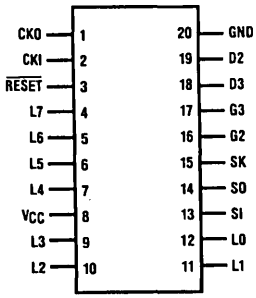
Note 3: Voltage change must be less than 0.5V in a 1 ms period.

Note 4: LED direct drive must not be used. Exercise great care not to exceed maximum device power dissipation limits when sourcing similar loads at high temperature.

Note 5: Variation due to the device included.

Connection Diagrams

COP422, COP322
DIP



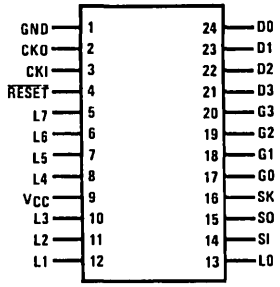
TL/DD/6921-4

Top View

Order Number COP322-XXX/N
or COP422-XXX/N
See NS Molded Package N20A

Order Number COP322-XXX/D
or COP422-XXX/D
See NS Hermetic Package D20A

COP421, COP321
DIP and SO Wide



TL/DD/6921-3

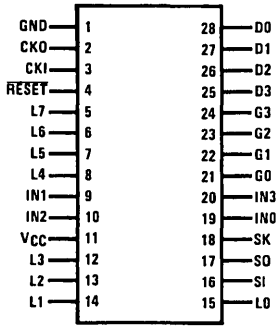
Top View

Order Number COP321-XXX/N
or COP421-XXX/N
See NS Molded Package N24A

Order Number COP321-XXX/D
or COP421-XXX/D
See NS Hermetic Package D24C

Order Number COP321-XXX/WM
or COP421-XXX/WM
See NS Surface Mount Package M24B

COP420, COP320
Dual-In-Line Package



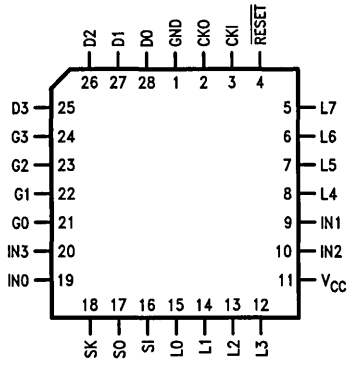
TL/DD/6921-2

Top View

Order Number COP320-XXX/N
or COP420-XXX/N
See NS Molded Package N28B

Order Number COP320-XXX/D
or COP320-XXX/D
See NS Hermetic Package D28C

PLCC



TL/DD/6921-31

Order Number COP320-XXX/V
or COP420-XXX/V
See NS PLCC Package V28A

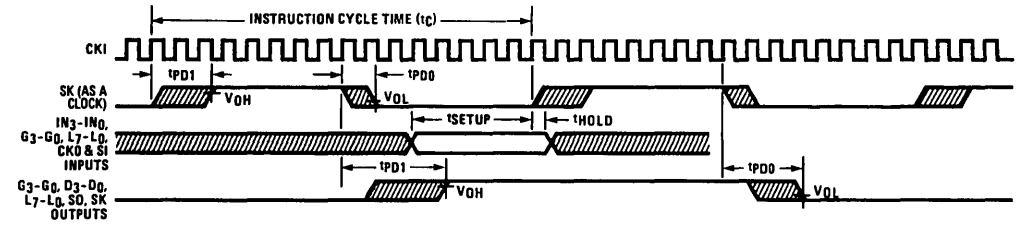
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs (COP420/320 only)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output

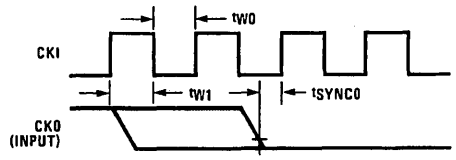
Pin	Description
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output (or general purpose input or RAM power supply)
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams



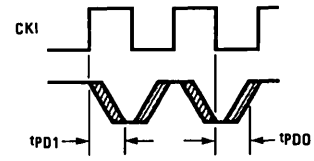
TL/DD/6921-5

FIGURE 3. Input/Output Timing Diagrams (Crystal Divide by 16 Mode)



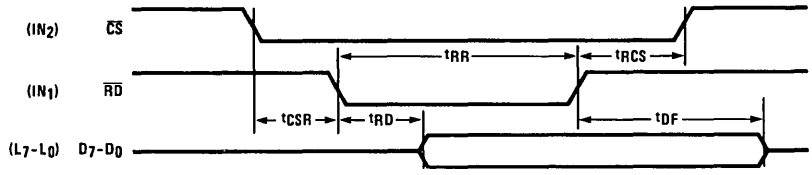
TL/DD/6921-6

FIGURE 3A. Synchronization Timing



TL/DD/6921-7

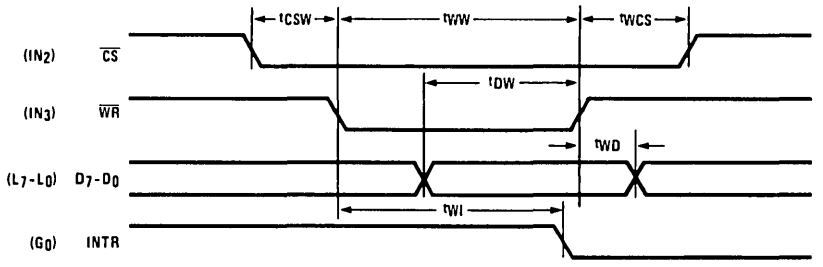
FIGURE 3B. CKO Output Timing



TL/DD/6921-8

FIGURE 4. MICROBUS Read Operation Timing

Timing Diagrams (Continued)



TL/DD/6921–9

FIGURE 5. MICROBUS Write Operation Timing

Functional Description COP420/COP421/COP422, COP320/COP321/COP322

For ease of reading this description, only COP420 and/or COP421 are referenced; however, all such references apply equally to the COP422, COP322, COP320 and/or COP321, respectively.

A block diagram of the COP420 is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024 byte ROM. As can be seen by an examination of the COP420/421 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 10-bit binary count value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit **B register** whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit **A register** (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load the input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

A **4-bit adder** performs the arithmetic and logic functions of the COP420/421, storing its results in A. It also outputs a carry bit to the 1-bit **C register**, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four **general-purpose inputs, IN_3–IN_0,** are provided; IN_1, IN_2 and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select and Write Strobe inputs, respectively, for use in MICROBUS applications.

The **D register** provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The **G register** contents are outputs to 4 general-purpose bidirectional I/O ports. G_0 may be mask-programmed as an output for MICROBUS applications.

The **Q register** is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.) With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The **8 L drivers**, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The **SIO register** functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers. For example of additional parallel output capacity see **Application #2.**

The XAS instruction copies C into the **SKL latch.** In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The **EN register** is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0 selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0" occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting let each instruction cycle time. The data present at DI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With the EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN enables SO as the output of the SIO shift register outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides summary of the modes associated with EN_3 and EN_1.

OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 8.*

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optional by 8).

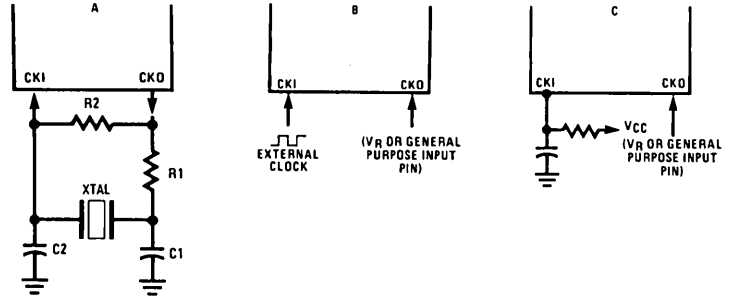
b. External Oscillator. CKI is an external clock input signal. The external frequency is divided by 16 (optional by 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R) of as a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available for non-timing functions.

Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)



TL/DD/6921–10

Crystal Oscillator **External Oscillator** **RC Controlled Oscillator**

Crystal Oscillator

Crystal Value	Component Values			
	R1(Ω)	R2(Ω)	C1(pF)	C2(pF)
4 MHz	4.7k	1M	22	22
3.58 MHz	3.3k	1M	22	27
2.09 MHz	8.2k	1M	47	33

RC Controlled Oscillator

R(kΩ)	C(pF)	Instruction Cycle Time (μs)
12	100	5 ±20%
6.8	220	5.3 ±23%
8.2	300	8 ±29%
22	100	8.6 ±16%

Note: 50 kΩ ≥ R ≥ 5 kΩ
360 pF ≥ C ≥50 pF

FIGURE 8. COP420/421/COP320/321 Oscillator

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP420/421 system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION (NOT AVAILABLE ON COP422)

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power up.
2. V_R must be within the operating range of the chip, and equal to V_{CC} ±1V during normal operation.
3. V_R must be ≥ 3.3V with V_{CC} off.

INTERRUPT

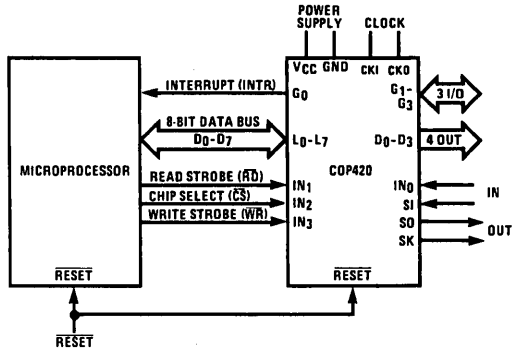
The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:
   1. EN_1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time,* the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)



TL/DD/6921-12

FIGURE 6. MICROBUS Option Interconnect

popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

MICROBUS™ INTERFACE

The COP420 has an option which allows it to be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN_1, IN_2 and IN_3 general purpose inputs become **MICROBUS compatible** read-strobe, chip-select, and write-strobe lines, respectively. IN_1 becomes $\overline{RD}$—a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN_2 becomes $\overline{CS}$—a logic "0" on this line selects the COP420 as the μP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN_3 becomes $\overline{WR}$—a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP420. G_0 becomes INTR a "ready" output, reset by a write pulse from the μP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP420.

This option has been designed for compatibility with National's MICROBUS—a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS National Publication.) The functioning and timing relationships between the COP420 signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection of the COP420 to the MICROBUS is shown in *Figure 6*.

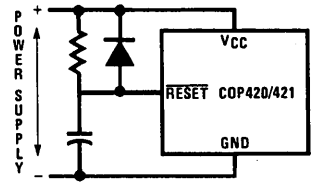
Note: TRI-STATE outputs must be used on L-port.

INITIALIZATION

The Reset Logic, internal to the COP420/421, will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{RESET}$ pin as shown below. The $\overline{RESET}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}.

Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/6921-13

FIGURE 7. Power-Up Clear Circuit

I/O OPTIONS

COP420/421 outputs have the following optional configurations, illustrated in *Figure 9a*:

a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. **Standard L**—same as **a.**, but may be disabled. Available on L outputs only.

e. **Open Drain L**—same as **b.**, but may be disabled. Available on L outputs only.

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

f. LED Direct Drive—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display.

g. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers.

COP420/COP421 inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

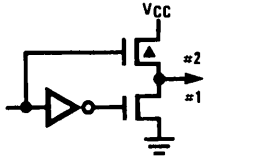
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 9b* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP420/421 system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1." The L outputs can be configured as in **d.**, **e.**, **f.** or **g.**

An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 9b,* device 2); however, when the L lines are used as input, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to logic "1".

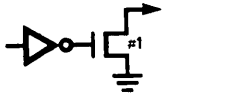
COP421

If the COP420 is bonded as a 24-pin device, it becomes the COP421, illustrated in *Figure 2,* COP420/421 Connection Diagrams. Note that the COP421 does not contain the four general purpose IN inputs (IN_3–IN_0). Use of this option precludes, of course, use of the IN options, interrupt feature, and the MICROBUS option which uses IN_1–IN_3. All other options are available for the COP421.



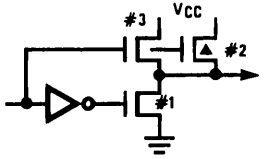
TL/DD/6921–14

a. Standard Output



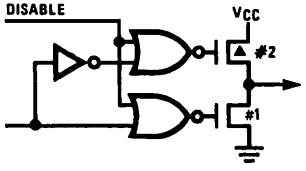
TL/DD/6921–15

b. Open-Drain Output



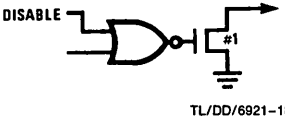
TL/DD/6921–16

c. Push-Pull Output



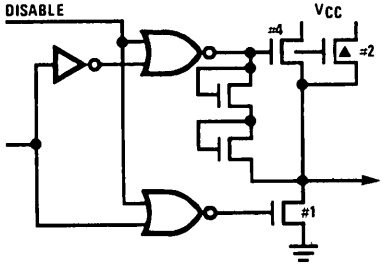
TL/DD/6921–17

d. Standard L Output



TL/DD/6921–18

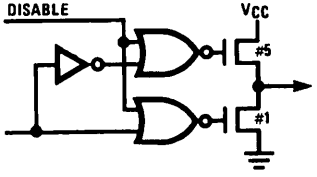
e. Open-Drain L Output



(▲ is Depletion Device)

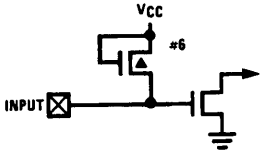
TL/DD/6921–19

f. LED (L Output)



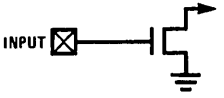
TL/DD/6921–20

g. TRI-STATE Push-Pull (L Output)



TL/DD/6921–21

h. Input with Load



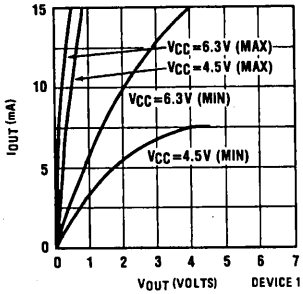
TL/DD/6921–22

I. HI-Z Input

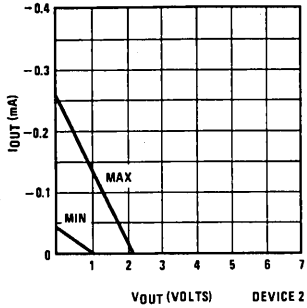
FIGURE 9a. Input/Output Configurations

Typical Performance Characteristics

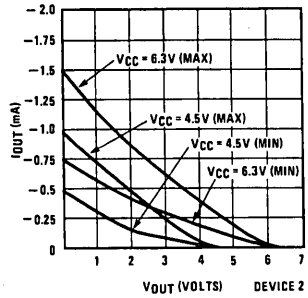
Output Sink Current



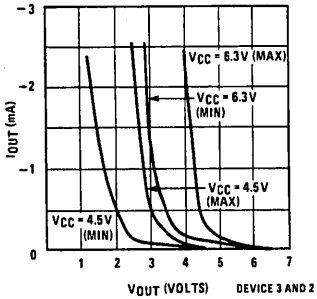
L Output Depletion Load OFF Source Current



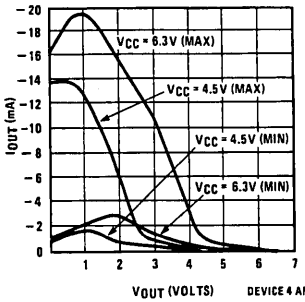
Standard Output Source Current



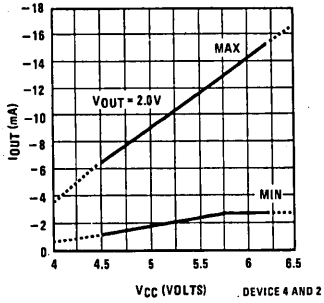
Push-Pull Source Current



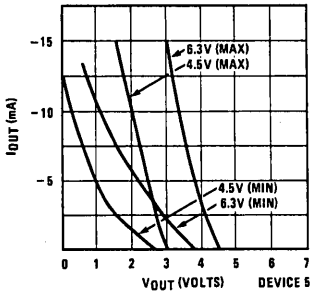
LED Output Source Current



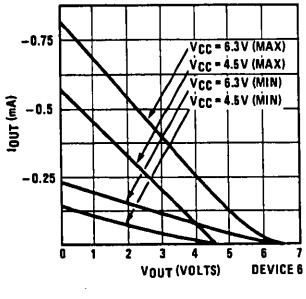
LED Output Direct LED Drive



TRI-STATE Output Source Current



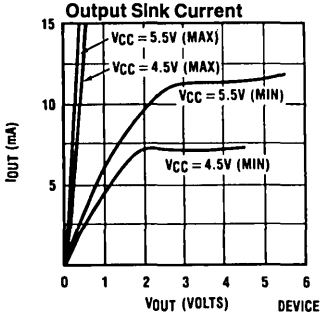
Input Load Source Current

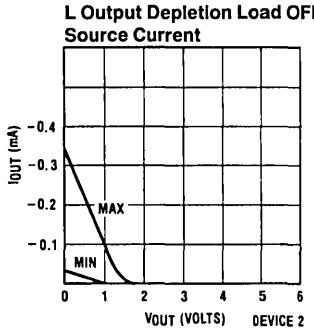


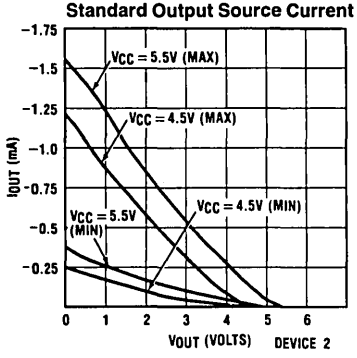
TL/DD/6921–23

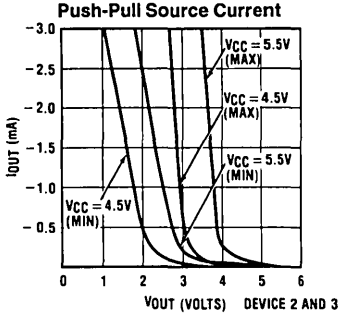
FIGURE 9b. COP420/COP421 Input/Output Characteristics

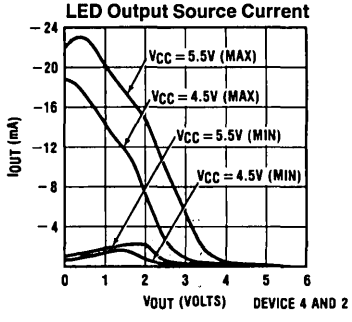
Typical Performance Characteristics (Continued)

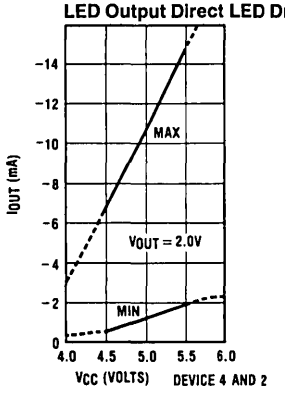


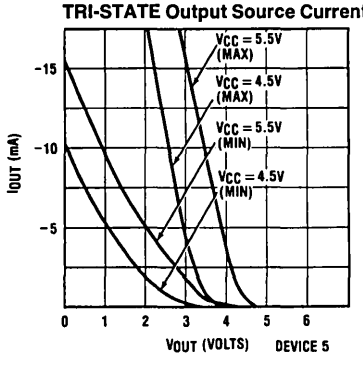


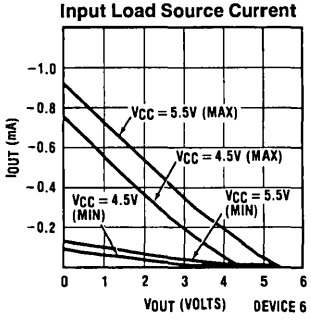












TL/DD/6921–24

FIGURE 9c. COP320/COP321 Input/Output Characteristics

Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP420/COP421/COP422 instruction set.

TABLE I. COP420/421/422/320/321/322 Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10 Subroutine Save Register A
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	10-bit Operand Field, 0-1023 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE II. COP420/421/422/320/321/322 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 | 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 | 0001	A + RAM(B) → A	None	Add RAM to A
ADT		4A	0100 | 1010	A + 10_{10} → A	None	Add Ten to A
AISC	y	5−	0101 | y	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CASC		10	0001 | 0000	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 | 0000	0 → A	None	Clear A
COMP		40	0100 | 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 | 0100	None	None	No Operation
RC		32	0011 | 0010	"0" → C	None	Reset C
SC		22	0010 | 0010	"1" → C	None	Set C
XOR		02	0000 | 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6–	0110 00 a_8	a → PC	None	Jump
		– –	$a_{7:0}$			
JP	a	– –	1 $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 4)
			or			
		– –	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → SA → SB → SC 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6–	0110 10 $a_{9:8}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
		– –	$a_{7:0}$			
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
		3C	0011 1100			
CQMA		33	0011 0011	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
		2C	0010 1100			
LD	r	–5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LDD	r,d	23	0010 0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		– –	00 r d			
LQID		BF	1011 1111	ROM($PC_{9:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	0 → $RAM(B)_1$		
	2	42	0100 0010	0 → $RAM(B)_2$		
	3	43	0100 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 1101	1 → $RAM(B)_1$		
	2	46	0100 0110	1 → $RAM(B)_2$		
	3	4B	0100 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	0010 0011	RAM(r,d) ⟷ A	None	Exchange A with RAM pointed to directly by r,d
		– –	10 r d			

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	−7	|00|r|0111|	RAM(B) ⟷ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	−4	|00|r|0100|	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	− −	|00|r|(d - 1)| (d = 0,9:15) or 33 |0011|0011| − − |10| r | d | (any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6−	|0011|0011| |0010| y |	y → EN	None	Load EN Immediate (Note 7)
XABR		12	|0001|0010|	A ⟷ Br (0,0 → A_3,A_2)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	G_0 = 0	
	1	11	|0001|0001|	2nd byte	G_1 = 0	
	2	03	|0000|0011|	2nd byte	G_2 = 0	
	3	13	|0010|0011|	2nd byte	G_3 = 0	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1$ = 0	
	2	03	|0000|0011|		$RAM(B)_2$ = 0	
	3	13	|0001|0011|		$RAM(B)_3$ = 0	
SKT		41	|0100|0001|		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	|0011|0011| |0010|1010|	G → A	None	Input G Ports to A
ININ		33 28	|0011|0011| |0010|1000|	IN → A	None	Input IN Inputs to A (Note 2)
INIL		33 29	|0011|0011| |0010|1001|	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	|0011|0011| |0010|1110|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	|0011|0011| |0011|1110|	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5–	|0011|0011| |0101| y |	y → G	None	Output to G Ports Immediate
OMG		33 3A	|0011|0011| |0011|1010|	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	|0100|1111|	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit register.

Note 2: The ININ instruction is not available on the COP421/COP321 and COP422/COP322 since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP420/421 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, $PC_{9:8}$, A, M. PC_9 and PC_8 are not affected by this instruction.

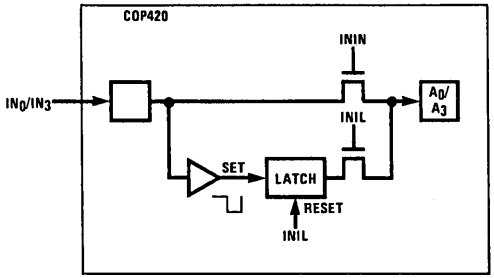
Note that JID requires 2 instruction cycles to execute.

Description of Selected Instructions (Continued)

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 10*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset.



TL/DD/6921–25

FIGURE 10

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the content of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP420/421 to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 131 kHz (crystal frequency ÷ 16) and the binary counter output pulse frequency will be 128 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 128 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP420/421 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instruction are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed except JID and LQID. LQID and JID take two cycle times if executed and one if skipped.

c. The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11 or 15 will access data in the next group of four pages.

Option List

The COP420/421/422 mask-programmable options are assigned numbers which correspond with the COP*420* pins.

The following is a list of COP420 options. When specifying a COP421 or COP422 chip, Options 9, 10, 19, 20 and 29 must all be set to zero. When specifying a COP422 chip, Options 21, 22, 27 and 28 must also be zero, and Option 2 must not be a 1. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground—no options available

Option 2: CKO Pin
- = 0: clock generator output to crystal
  0 not available if option 3 = 4 or 5)
- = 1: Pin is RAM power supply (V_R) input
  (Not available on COP422/COP322)
- = 2: general purpose input with load device
- = 4: general purpose Hi Z input

Option 3: CKI Input
- = 0: crystal input devided by 16
- = 1: crystal input divided by 8
- = 2: TTL external clock input divided by 16
- = 3: TTL external clock input divided by 8
- = 4: single-pin RC controlled oscillator (÷4)
- = 5: Schmitt trigger clock input (÷4)

Option 4: RESET Pin
- = 0: load devices to V_{CC}
- = 1: Hi-Z input

Option 5: L_7 Driver
- = 0: Standard output *(Figure 9D)*
- = 1: Open-Drain output (E)
- = 2: LED direct drive output (F)
- = 3: TRI-STATE push-pull output (G)

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: IN_1 Input
- = 0: load devices to V_{CC} (H)
- = 1: Hi-Z input (I)

Option 10: IN_2 Input
same as Option 9

Option 11 = 0: V_{CC} Pin—no options available

Option 12: L_3 Driver
same as Option 5

Option 13: L_2 Driver
same as Option 5

Option 14: L_1 Driver
same as Option 5

Option 15: L_0 Driver
same as Option 5

Option 16: SI Input
same as Option 9

Option 17: SO Driver
- = 0: standard output (A)
- = 1: open-drain output (B)
- = 2: push-pull output (C)

Option 18: SK Driver
same as Option 17

Option 19: IN_0 Input
same as Option 9

Option 20: IN_3 Input
same as Option 9

Option 21: G_0 I/O Port
- = 0: Standard output (A)
- = 1: Open-Drain output (B)

Option 22: G_1 I/O Port
same as Option 21

Option 23: G_2 I/O Port
same as Option 21

Option 24: G_3 I/O Port
same as Option 21

Option 25: D_3 Output
- = 0: Standard output (A)
- = 1: Open-Drain output (B)

Option 26: D_2 Output
same as Option 25

Option 27: D_1 Output
same as Option 25

Option 28: D_0 Output
same as Option 25

Option 29: COP Function
- = 0: normal operation
- = 1: MICROBUS option

Option 30: COP Bonding
- = 0: COP420 (28-pin device)
- = 1: COP421 (24-pin device)
- = 2: 28- and 24-pin device
- = 3: COP422 (20-pin device)
- = 4: 28- and 20-pin device
- = 5: 24- and 20-pin device
- = 6: 28-, 24- and 20-pin device

Option 31: In Input Levels
- = 0: normal input levels
- = 1: Higher voltage input levels
  ("0" = 1.2V, "1" = 3.6V)

Option 32: G Input Levels
same as Option 31

Option 33: L Input Levels
same as Option 31

Option 34: CKO Input Levels
same as Option 31

Option 35: SI Input Levels
same as Option 31

Option List (Continued)

COP OPTION LIST

The following option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE = 0 IS: GROUND PIN
OPTION 2 VALUE = _____ IS: CKO PIN
OPTION 3 VALUE = _____ IS: CKI INPUT
OPTION 4 VALUE = _____ IS: RESET INPUT
OPTION 5 VALUE = _____ IS: L_7 DRIVER
OPTION 6 VALUE = _____ IS: L_6 DRIVER
OPTION 7 VALUE = _____ IS: L_5 DRIVER
OPTION 8 VALUE = _____ IS: L_4 DRIVER
OPTION 9 VALUE = _____ IS: IN1 INPUT
OPTION 10 VALUE = _____ IS: IN2 INPUT
OPTION 11 VALUE = _____ IS: VCC PIN
OPTION 12 VALUE = _____ IS: L_3 DRIVER
OPTION 13 VALUE = _____ IS: L_2 DRIVER
OPTION 14 VALUE = _____ IS: L_1 DRIVER
OPTION 15 VALUE = _____ IS: L_0 DRIVER
OPTION 16 VALUE = _____ IS: SI INPUT
OPTION 17 VALUE = _____ IS: SO DRIVER
OPTION 18 VALUE = _____ IS: SK DRIVER
OPTION 19 VALUE = _____ IS: IN_0 INPUT
OPTION 20 VALUE = _____ IS: IN_3 INPUT
OPTION 21 VALUE = _____ IS: G_0 I/O PORT
OPTION 22 VALUE = _____ IS: G_1 I/O PORT
OPTION 23 VALUE = _____ IS: G_2 I/O PORT
OPTION 24 VALUE = _____ IS: G_3 I/O PORT
OPTION 25 VALUE = _____ IS: D_3 OUTPUT
OPTION 26 VALUE = _____ IS: D_2 OUTPUT
OPTION 27 VALUE = _____ IS: D_1 OUTPUT
OPTION 28 VALUE = _____ IS: D_0 OUTPUT
OPTION 29 VALUE = _____ IS: COP FUNCTION
OPTION 30 VALUE = _____ IS: COP BONDING
OPTION 31 VALUE = _____ IS: IN INPUT LEVELS
OPTION 32 VALUE = _____ IS: G INPUT LEVELS
OPTION 33 VALUE = _____ IS: L INPUT LEVELS
OPTION 34 VALUE = _____ IS: CKO INPUT LEVELS
OPTION 35 VALUE = _____ IS: SI INPUT LEVELS

TEST MODE (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP420. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATION # 1: COP420 General Controller

Figure 8 shows an interconnect diagram for a COP420 used as a general controller. Operation of the system is as follows:

1. The L_7–L_0 outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display.
2. The D_3–D_0 outputs drive the digits of the mulitplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN_3–IN_0 inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a V_R RAM power supply pin. RAM data integrity is thereby assured when the main power supply is shut down (see RAM Keep-Alive option description).
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G_3–G_0) are available for use as required by the user's application.

APPLICATION #2: MUSICAL ORGAN AND MUSIC BOX

Play Mode: Twenty-five musical keys and 25 LEDs are provided to denote F to F with half notes in between. All the keys and LEDs are directly detected and driven by the microprocessor. Depression of the key will give the corresponding musical note and light up the corresponding LED.

Clear: Memory is provided to store a played tune. Depression of the CLEAR key erases the memory and the microprocessor is ready to store new musical notes. A maximum of 28 notes can be stored where each note can be of one to eight musical beats. (Two bytes of memory are required to store one musical note. Any note longer than eight musical beats will require additional memory space for storage.)

Playback: Depression of this button will playback the tune stored in the memory since last "clear."

Preprogrammed Tunes: There are ten preprogrammed tunes (each has an average of 55 notes) masked in the chip. Any tune can be recalled by depressing the "Tune Button" followed by the corresponding "Sharp Key."

Learn Mode: This mode is for the player to learn the ten preprogrammed tunes. By pressing the "Learn Button" followed by the corresponding "Sharp Key," the LEDs will be lighted up one by one to indicate the notes of the selected tune. The LED will remain "on" until the player presses the correct musical key; the LED for the next note will then be lighted up.

Pause: In addition to the 25 musical keys, there is a special pause key. The depression of this key generates a blank note to the memory.

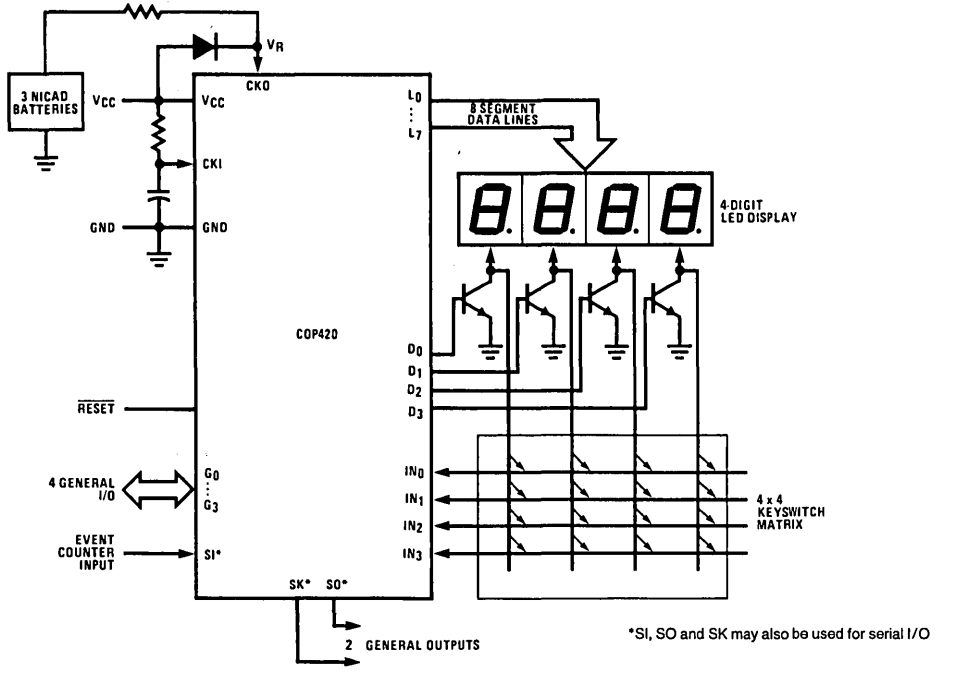
Note: In the Learn Mode when playing "Oh Susanna," the pause key must be used.

Tempo: This is a control input to the musical beat time oscillator for varying the speed of the musical tunes.

Vibrato: This is a switch control to vary the frequency vibration of the note.

Tunes Listing: The following is a listing of the ten preprogrammed tunes: 1) Jingle Bells, 2) Twinkle, Twinkle Little Star, 3) Happy Birthday, 4) Yankee Doodle, 5) Silent Night, 6) This Old Man, 7) London Bridge Is Falling Down, 8) Auld Lang Syne, 9) Oh Susanna, 10) Clementine.

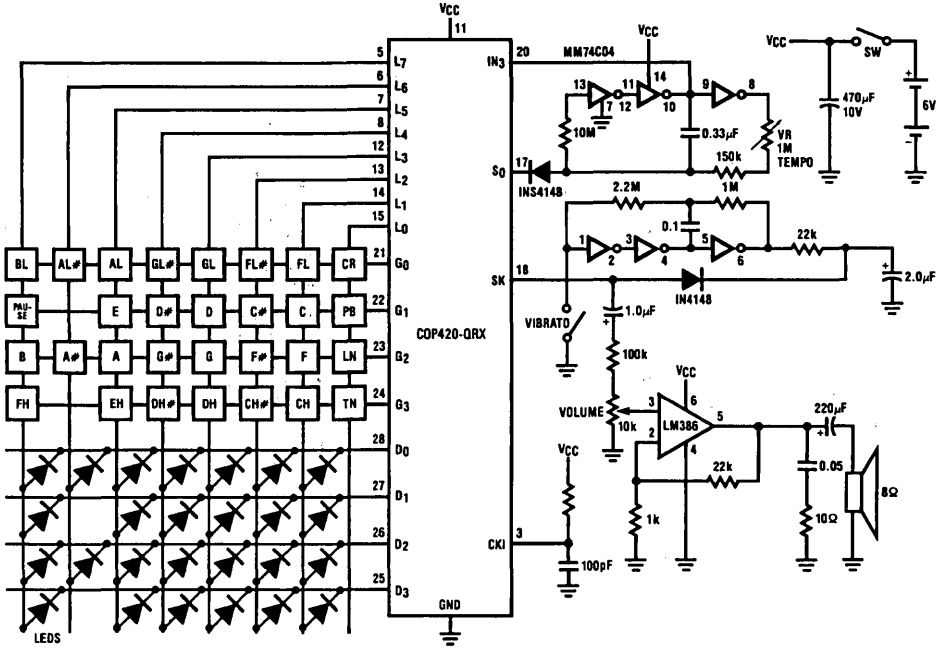
Typical Applications



TL/DD/6921–26

FIGURE 11. COP420 Keyboard Display Interface

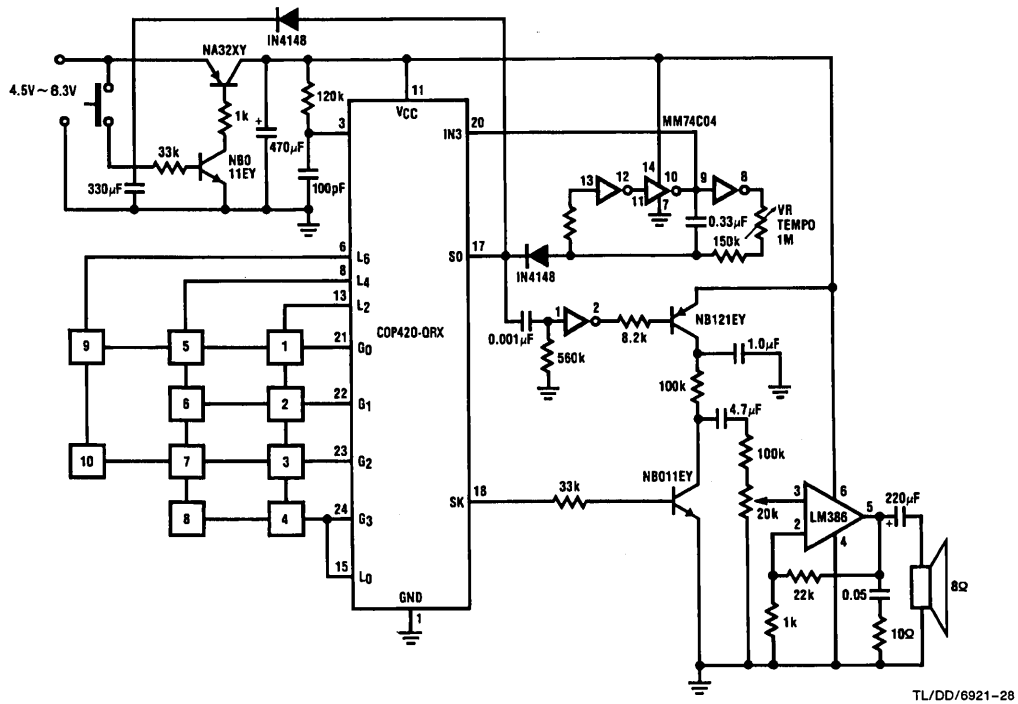
Circuit Diagram of COP420 Musical Organ



TL/DD/6921–27

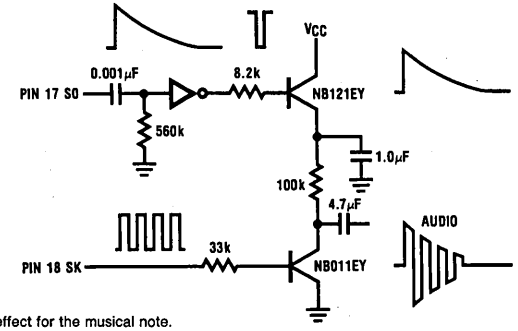
Typical Applications (Continued)

Music Box Application with Direct Key Access



TL/DD/6921-28

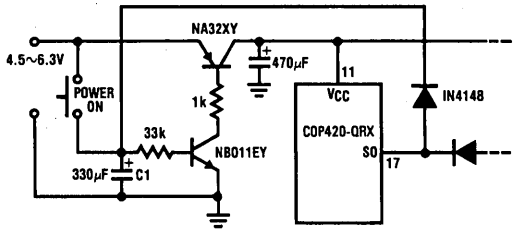
Bell Sound Circuit



This additional circuit provides tinkling effect for the musical note.

TL/DD/6921-29

Auto Power Shut-Off Circuit



TL/DD/6921-30

This circuit automatically turns off the musical organ if none of the keys are pressed within approximately 30 seconds.

National Semiconductor

COP420L/COP421L/COP422L/COP320L/COP321L/ COP322L Single-Chip N-Channel Microcontrollers

General Description

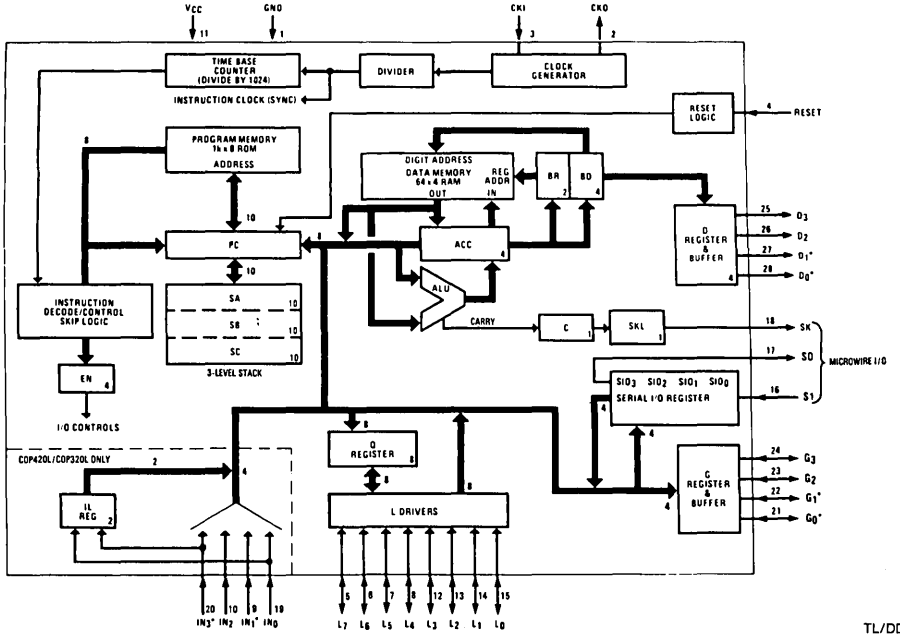
The COP420L, COP421L, COP422L, COP320L, COP321L, and COP322L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These controller oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP421L and COP422L are identical to the COP420L, but with 19 and 15 I/O lines, respectively, instead of 23. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

The COP320L, COP321L, and COP322L are exact functional equivalents, but extended temperature range versions, of the COP420L, COP421L, and COP422L respectively.

Features

- Low cost
- Powerful instruction set
- 1k x 8 ROM, 64 x 4 RAM
- 23 I/O lines (COP420L)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (9 mA max)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device— COP320L/COP321L/COP322L (−40°C to +85°C)
- Wider supply range (4.5V–9.5V) optionally available

Block Diagram



TL/DD/8825–1

*Not available on COP422L/COP322L

FIGURE 1

COP420L/COP421L/COP422L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Power Dissipation	
COP420L/COP421L	0.75W at 25°C 0.4W at 70°C
COP422L	0.65W at 25°C 0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Optional Operating Voltage (V_{CC})		4.5	9.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		9	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷32, ÷16, ÷8)				
Logic High (V_{IH}) V_{CC} = Max		3.0		V
Logic High (V_{IH})				
V_{CC} = 5V ±5%		2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 3)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	with TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.8	V
Logic High	with High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 ma		0.4	V
CMOS Operation (Note 2)	V_{CC} = 4.5V			
Logic High	I_{OH} = −10 μA	V_{CC} −1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.8V for normal operation.

COP420L/COP421L/COP422L

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	1.8		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	1.2		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.9		mA
L_0–L_7 Outputs and Standard G_0–G_3, D_0–D_3 Outputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with High Current Options (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 1.0V$	15		mA
	$V_{CC} = 6.3V, V_{OL} = 1.0V$	11		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7.5		mA
G_0–G_3 and D_0–D_3 Outputs with Very High Current Options (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 1.0V$	30		mA
	$V_{CC} = 6.3V, V_{OL} = 1.0V$	22		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	15		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−140	−800	μA
	$V_{CC} = 6.3V, V_{OH} = 2.0V$	−75	−480	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−30	−250	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 4.75V$	−1.4		mA
	$V_{CC} = 6.3V, V_{OH} = 2.4V$	−1.4		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−1.5	−18	mA
	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.5	−13	mA
LED Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−3.0	−35	mA
	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−3.0	−25	mA
TRI-STATE Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 5.5V$	−0.75		mA
	$V_{CC} = 6.3V, V_{OH} = 3.2V$	−0.8		mA
	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 5.5V$	−1.5		mA
	$V_{CC} = 6.3V, V_{OH} = 3.2V$	−1.6		mA
	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−140	μA
CKO Output RAM Power Supply Option Power Requirement	$V_R = 3.3V$		3.0	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

COP320L/COP321L/COP322L

Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	
COP320L/COP321L	0.75W at 25°C
	0.4W at 70°C
	0.25W at 85°C
COP322L	0.65W at 25°C
	0.20W at 70°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Optional Operating Voltage (V_{CC})		4.5	7.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		11	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH}) V_{CC} = Max		3.0		V
Logic High (V_{IH})				
V_{CC} = 5V ±5%		2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 3)	2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	with TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	with High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	µA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 µA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)	V_{CC} = 4.5V			
Logic High	I_{OH} = −10 µA	V_{CC} −1		V
Logic Low	I_{OL} = +10 µA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

COP320L/COP321L/COP322L

DC Electrical Characteristics

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 7.5V, V_{OL} = 0.4V$	1.4		mA
	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0–L_7 Outputs and Standard	$V_{CC} = 7.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 7.5V, V_{OL} = 1.0V$	12		mA
High Current Options (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 1.0V$	9		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 7.5V, V_{OL} = 1.0V$	24		mA
Very High Current Options (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 1.0V$	18		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	14		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−100	−900	μA
All Outputs (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	μA
Push-Pull Configuration	$V_{CC} = 7.5V, V_{OH} = 3.75V$	−0.85		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−1.4	−27	mA
Outputs, Low Current	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.4	−17	mA
Driver Option (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−0.7	−15	mA
LED Configuration, L_0–L_7	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−2.7	−54	mA
Outputs, High Current	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−2.7	−34	mA
Driver Option (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.4	−30	mA
TRI-STATE Configuration,	$V_{CC} = 7.5V, V_{OH} = 4.0V$	−0.7		mA
L_0–L_7 Outputs, Low	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−0.6		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration,	$V_{CC} = 7.5V, V_{OH} = 4.0V$	−1.4		mA
L_0–L_7 Outputs, High	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−1.2		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−200	μA
CKO Output				
RAM Power Supply Option	$V_R = 3.3V$		4.0	mA
Power Requirement				
TRI-STATE Output Leakage Current		−5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

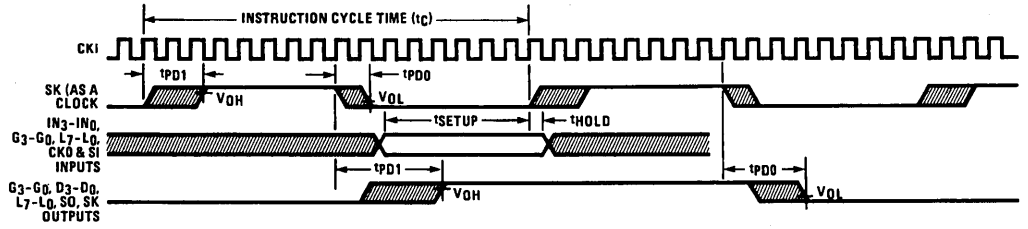
AC Electrical Characteristics

COP420L/COP421L/COP422L: 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 9.5V unless otherwise noted
COP320L/COP321L/COP322L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time—t_C		16	40	μs
CKI				
Input Frequency—f_I	÷32 Mode	0.8	2.0	MHz
	÷16 Mode	0.4	1.0	MHz
	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 2 MHz		120	ns
Fall Time			80	ns
CKI Using RC (÷4)	R = 56 kΩ ±5% C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
INPUTS:				
IN_3–IN_0, G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

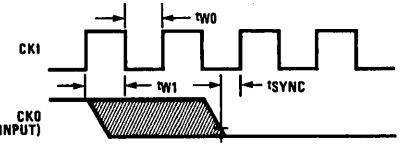
Note 1: Variation due to the device included.

Timing Diagrams



TL/DD/8825–5

FIGURE 3. Input/Output Timing Diagrams (Crystal Divide-by-16 Mode)

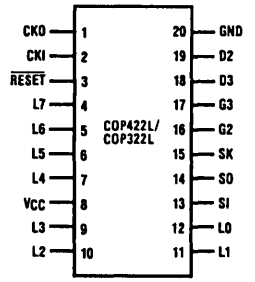


TL/DD/8825–6

FIGURE 3a. Synchronization Timing

Connection Diagrams

SO Wide and DIP



TL/DD/8825-4

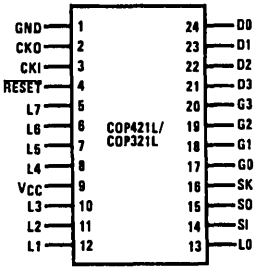
Top View

Order Number COP422L-XXX/N
or COP322L-XXX/N
See NS Molded Package Number N24A

Order Number COP322L-XXX/D
or COP422L-XXX/D
See NS Hermetic Package Number D20A

Order Number COP322L-XXX/WM
or COP422L-XXX/WM
See NS Surface Mount Package Number M20B

SO Wide and DIP



TL/DD/8825-3

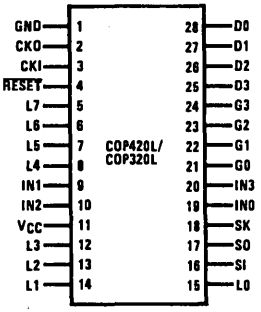
Top View

Order Number COP421L-XXX/N
or COP321L-XXX/N
See NS Molded Package Number N20A

Order Number COP321L-XXX/D
or COP421L-XXX/D
See NS Hermetic Package Number D24C

Order Number COP321L-XXX/WM
or COP421L-XXX/WM
See NS Surface Mount Package Number M24B

Dual-In Line Package



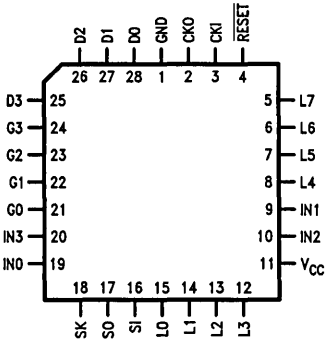
TL/DD/8825-2

Top View

Order Number COP420L-XXX/N
or COP320L-XXX/N
See NS Molded Package Number N28B

Order Number COP320L-XXX/D
or COP420L-XXX/D
See NS Hermetic Package Number D28C

PLCC



TL/DD/8825-27

Order Number COP320L-XXX/V
or COP420L-XXX/V
See NS PLCC Package Number V28A

FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs (COP420L only)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output (or general purpose input, RAM power supply or SYNC input)
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground

Functional Description

For ease of reading this description, only COP420L and/or COP421L are referenced; however, all such references apply also to COP320L, COP321L, COP322L, or COP422L.

A block diagram of the COP420L is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024 byte ROM. As can be seen by an examination of the COP420L/421L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 10-bit binary count value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions is based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP420/421L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunctions with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four general-purpose inputs, IN_3–IN_0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers. For example of additional parallel output capacity see Application #2.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables

Functional Description (Continued)

the L drivers, placing the L I/O ports in a high-impedance input state.

4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides a summary of the modes associated with EN_3 and EN_0.

Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once aknowledged as explained below, pushes the next sequential program counter address (PC + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:
   1. EN_1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time,* the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt servicing routine since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

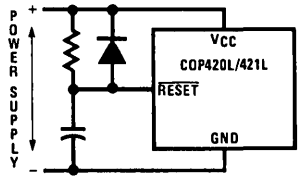
e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

Power-Up Clear Circuit



TL/DD/8825-7

RC ≥ 5 × Power Supply Rise Time

Functional Description (Continued)

OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 4.*

a. **Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32 (optional by 16 or 8).

b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 32 (optional by 16 or 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R) or as a general purpose input.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or as a general purpose input.

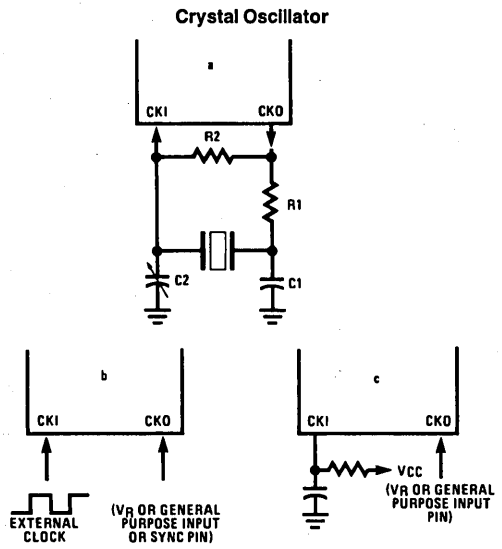
CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP420L/421L system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION (Not available on COP422L)

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. During normal operation V_R must be within the operating range of the chip, with $(V_{CC} - 1) \leq V_R \leq V_{CC}$.
3. V_R must be $\geq$ 3.3V with V_{CC} off.



TL/DD/8825–8

Crystal Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220
2.097 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (μs)
51	100	19 ±15%
82	56	19 ± 13%

Note: 200k ≥ R ≥ 25k
360 pF ≥ C ≤ 50 pF

FIGURE 4. COP420L/421L Oscillator

Functional Description (Continued)

I/O OPTIONS

COP420L/421L outputs have the following optional configurations, illustrated in *Figure 5*:

a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. **Standard L**—same as **a.**, but may be disabled. Available on L outputs only.

e. **Open Drain L**—same as **b.**, but may be disabled. Available on L outputs only.

f. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

g. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

COP420L/COP421L inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP420L/421L system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in **d.**, **e.**, **f.** or **g.**

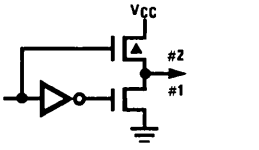
An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6,* device 2); however, when the L lines are used as inputs, the disabled depletion device *cannot* be relied on to source sufficient current to pull an input to a logic 1.

COP421L

If the COP420L is bonded as a 24-pin device, it becomes the COP421L, illustrated in *Figure 2,* COP420L/421L Connection Diagrams. Note that the COP421L does not contain the four general purpose IN inputs (IN_3–IN_0). Use of this option precludes, of course, use of the IN options and the interrupt feature. All other options are available for the COP421L.

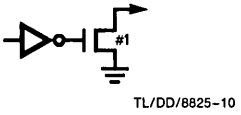
COP422L

If the COP421L is bonded as a 20-pin device, it becomes the COP422L, as illustrated in *Figure 2.* Note that the COP422L contains all the COP421L pins except D_0, D_1, G_0, and G_1. COP422L also does not allow RAM power supply input as a valid CKO pin option.



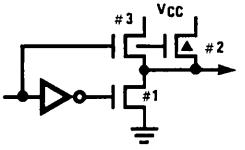
TL/DD/8825-9

a. Standard Output



TL/DD/8825-10

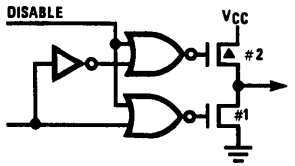
b. Open-Drain Output



TL/DD/8825-11

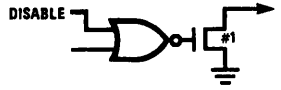
c. Push-Pull Output

Functional Description (Continued)



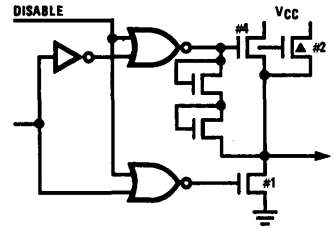
TL/DD/8825-12

d. Standard L Output



TL/DD/8825-13

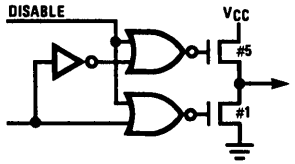
e. Open-Drain L Output



TL/DD/8825-14

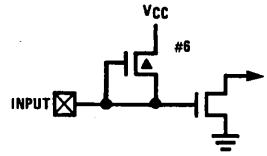
(▲ is Depletion Device)

f. LED (L Output)



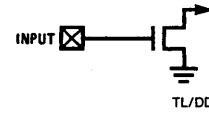
TL/DD/8825-15

g. TRI-STATE Push-Pull (L Output)



TL/DD/8825-16

h. Input with Load



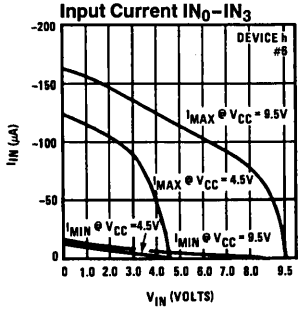
TL/DD/8825-17

I. HI-Z Input

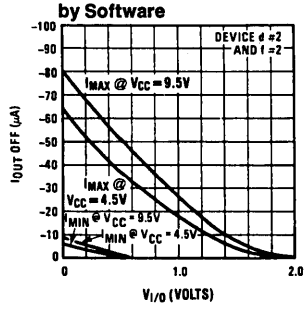
FIGURE 5. Output Configurations

Typical Performance Characteristics

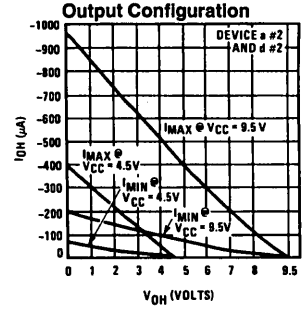
Input Current IN_0–IN_3



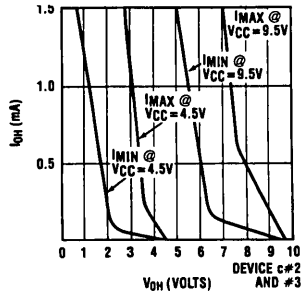
Input Current for L_0–L_7 when Output Programmed OFF by Software



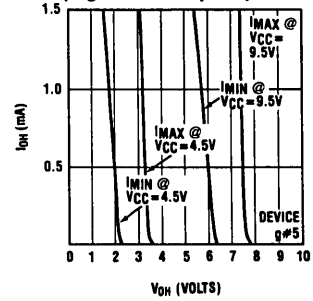
Source Current for Standard Output Configuration



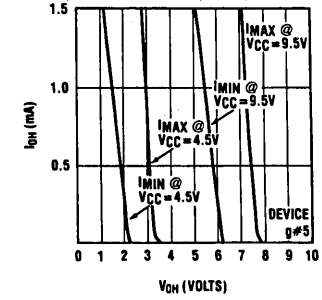
Source Current for SO and SK in Push-Pull Configuration



Source Current for L_0–L_7 in TRI-STATE Configuration (High Current Option)



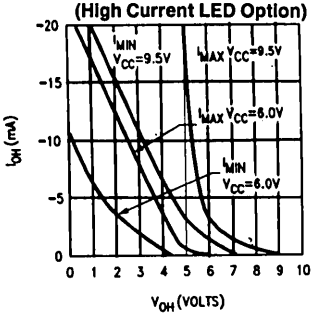
Source Current for L_0–L_7 in TRI-STATE Configuration (Low Current Option)



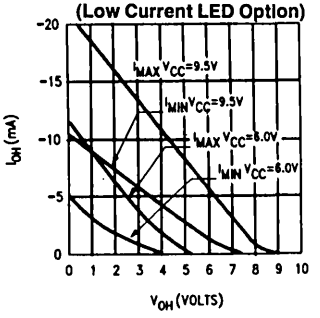
TL/DD/8825-18

Typical Performance Characteristics (Continued)

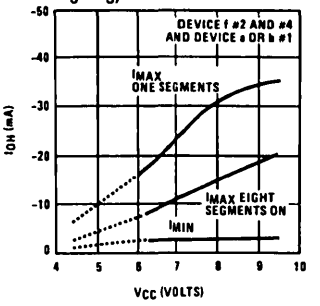
LED Output Source Current (High Current LED Option)



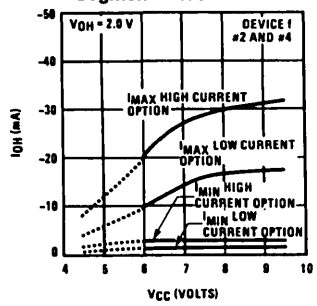
LED Output Source Current (Low Current LED Option)



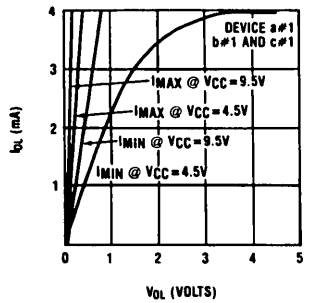
LED Output Direct Segment and Digit Drive (High Current Options on L_0–L_7; Very High Current Options on D_0–D_3 or G_0–G_3)



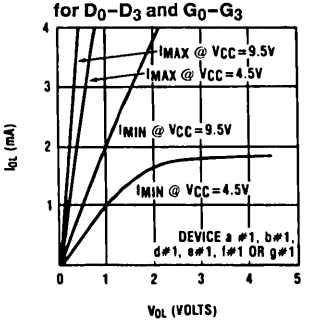
LED Output Direct Segment Drive



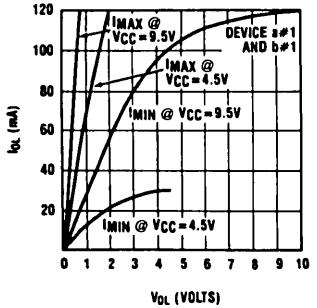
Output Sink Current for SO and SK



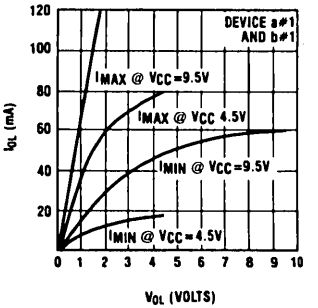
Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3



Output Sink Current for G_0–G_3 and D_0–D_3 with Very High Current Option



Output Sink Current for G_0–G_3 and D_0–D_3 (High Current Option)

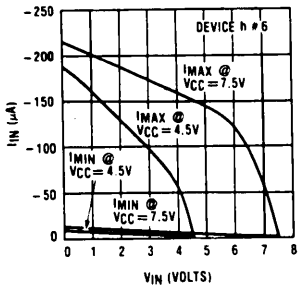


TL/DD/8825–19

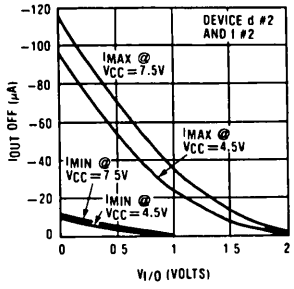
FIGURE 6. COP420L/COP421L/COP422L Input/Output Characteristics

Typical Performance Characteristics (Continued)

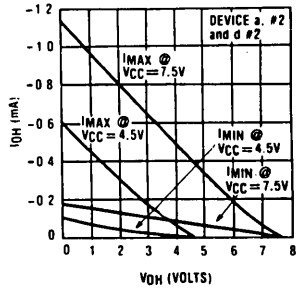
Input Current IN_0–IN_3



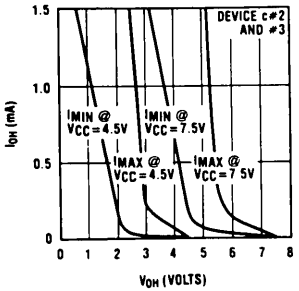
Input Current for L_0–L_7 when Output Programmed OFF by Software



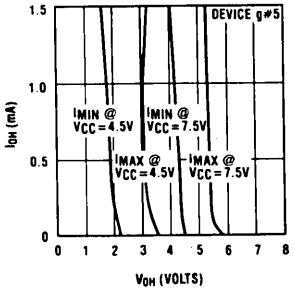
Source Current for Standard Configuration



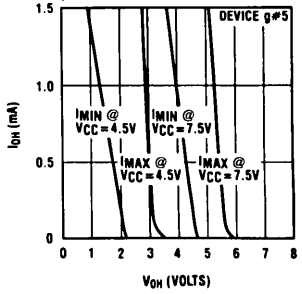
Source Current for SO and SK in Push-Pull Configuration



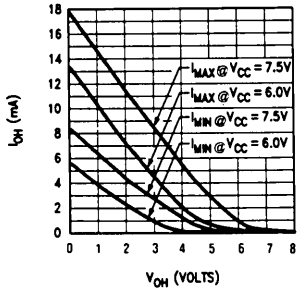
Source Current for L_0–L_7 in TRI-STATE Configuration (High Current Option)



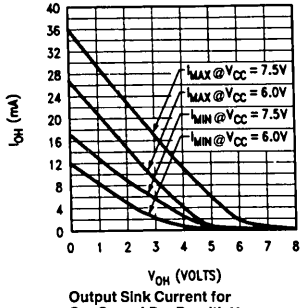
Source Current for L_0–L_7 in TRI-STATE Configuration (Low Current Option)



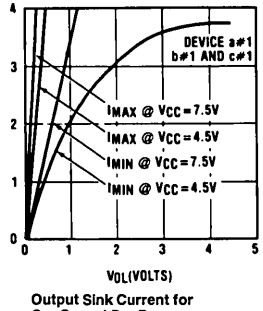
LED Output Source Current (Low Current LED Option)



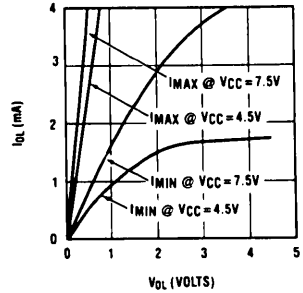
LED Output Source Current (High Current LED Option)



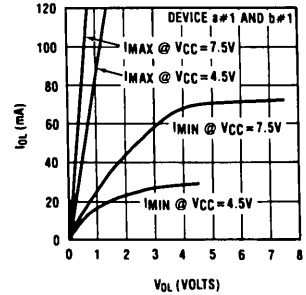
Output Sink Current for SO and SK



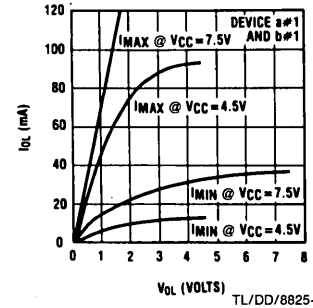
Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3



Output Sink Current for G_0–G_3 and D_0–D_3 with Very High Current Option



Output Sink Current for G_0–G_3 and D_0–D_3 (High Current Option)



TL/DD/8825–20

FIGURE 7. COP320L/DOP321L/COP322L Input/Output Characteristics

COP420L/COP421L Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410L/411L instruction set.

TABLE I. COP420L/421L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit Latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	10-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	10-bit Operand Field, 0–1023 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The ones complement of A
⊕	Exclusive-OR
:	Range of values

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Compliment and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM ($PC_{9:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– – –	|0110|00|$a_{9:8}$| |$a_{7:0}$|	a → PC	None	Jump
JP	a	– –	|1| $a_{6:0}$ | (pages 2,3 only) or |11| $a_{5:0}$ | (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 4)
JSRP	a	– –	|10| $a_{5:0}$ |	PC + 1 → SA → SB → SC 0010 → $PC_{9:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	|0110|10|$a_{9:8}$| |$a_{7:0}$|	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011|0011 0011|1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011|0011 0010|1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00|r|0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 – –	0010|0011 00| r | d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011|1111	ROM($PC_{9:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100|1100 0100|0101 0100|0010 0100|0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100|1101 0100|1101 0100|0110 0100|1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111| y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00|r|0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 – –	0010|0011 10| r | d	RAM(r,d) ⟷ A	None	Exchange A with RAM pointed to directly by (r,d)
XDS	r	–7	00|r|0111	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	00|r|0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	--	00 | r | (d−1) (d=0,9:15) or	r,d → B	Skip until not an LBI	Load B Immediate with r,d (Note 6)
		33	0011 | 0011			
		--	10 | r | d (any d)			
LEI	y	33	0011 | 0011	y → EN	None	Load EN Immediate (Note 7)
		6-	0110 | y			
XABR		12	0001 | 0010	A ⟷ Br (0,0 → A_3,A_2)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 | 0011		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
		21	0010 | 0001			
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	G_0 = 0	
	1	11	0001 | 0001		G_1 = 0	
	2	03	0000 | 0011		G_2 = 0	
	3	13	0001 | 0011		G_3 = 0	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1$ = 0	
	2	03	0000 | 0011		$RAM(B)_2$ = 0	
	3	13	0001 | 0011		$RAM(B)_3$ = 0	
SKT		41	0100 | 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 | 0011	G → A	None	Input G Ports to A
		2A	0010 | 1010			
ININ		33	0011 | 0011	IN → A	None	Input IN Inputs to A (Note 2)
		28	0010 | 1000			
INIL		33	0011 | 0011	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
		29	0010 | 1001			
INL		33	0011 | 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 | 1110	$L_{3:0}$ → A		
OBD		33	0011 | 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 | 1110			
OGI	y	33	0011 | 0011	y → G	None	Output to G Ports Immediate
		5–	0101 | y			
OMG		33	0011 | 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is only available on the 28-pin COP420L as the other devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP420L/421L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, $PC_{9:8}$, A, M. PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

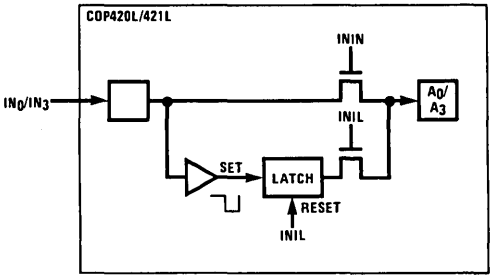
Description of Selected Instructions (Continued)

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 8*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction. IL latches are *not cleared* on reset.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note the LQID takes two instruction cycle times to execute.



TL/DD/8825–21

FIGURE 8. INIL Hardware Implementation

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP420L/421L to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷ 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP420L/421L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, or 15 will access data in the next group of four pages.

Option List

The COP420L/421L mask-programmable options are assigned numbers which correspond with the COP*420L* pins.

The following is a list of COP420L options. When specifying a COP421L chip, Options 9, 10, 19, and 20 must all be set to zero. When specifying a COP422L chip, options 9, 10, 19, and 20 must all be set to zero; options 21 and 22 may not be set to one, three or five; and option 2 may not be set to one. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

The Option Table should be copied and sent in with your EPROM or disc.

Option 1 = 0: Ground Pin—no options available

Option 2: CKO Output
- = 0: clock generator output to crystal/resonator (0 not allowable value if Option 3 = 3)
- = 1: pin is RAM power supply (V_R) input (not available on the COP422L)
- = 2: general purpose input with load device to V_{CC}
- = 3: general purpose input, Hi-Z

Option 3: CKI Input
- = 0: oscillator input divided by 32 (2 MHz max.)
- = 1: oscillator input divided by 16 (1 MHz max.)
- = 2: oscillator input divided by 8 (500 kHz max.)
- = 3: single-pin RC controlled oscillator (÷4)
- = 4: Schmitt trigger clock input (÷4)

Option 4: RESET Input
- = 0: load device to V_{CC}
- = 1: Hi-Z Input

Option 5: L_7 Driver
- = 0: Standard output
- = 1: Open-drain output
- = 2: High current LED direct segment drive output
- = 3: High current TRI-STATE push-pull output
- = 4: Low-current LED direct segment drive output
- = 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: IN_1 Input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN_2 Input
same as Option 9

Option 11: V_{CC} pin
- = 0: Standard V_{CC}
- = 1: Optional higher voltage V_{CC}

Option 12: L_3 Driver
same as Option 5

Option 13: L_2 Driver
same as Option 5

Option 14: L_1 Driver
same as Option 5

Option 15: L_0 Driver
same as Option 5

Option 16: SI Input
same as Option 9

Option 17: SO Driver
- = 0: standard output
- = 1: open-drain output
- = 2: push-pull output

Option 18: SK Driver
same as Option 17

Option 19: IN_0 Input
same as Option 9

Option 20: IN_3 Input
same as Option 9

Option 21: G_0 I/O Port
- = 0: very-high current standard output
- = 1: very-high current open-drain output
- = 2: high current standard output
- = 3: high current open-drain output
- = 4: standard LSTTL output (fanout = 1)
- = 5: open-drain LSTTL output (fanout = 1)

Option 22: G_1 I/O Port
same as Option 21

Option 23: G_2 I/O Port
same as Option 21

Option 24: G_3 I/O Port
same as Option 21

Option 25: D_3 Output
same as Option 21

Option 26: D_2 Output
same as Option 21

Option 27: D_1 Output
same as Option 21

Option 28: D_0 Output
same as Option 21

Option 29: L Input Levels
- = 0: standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
- = 1: higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 30: IN Input Levels
same as Option 29

Option 31: G Input Levels
same as Option 29

Option 32: SI Input Levels
same as Option 29

Option 33: $\overline{\text{RESET}}$ Input
- = 0: Schmitt trigger input
- = 1: standard TTL input levels
- = 2: higher voltage input levels

Option 34: CKO Input Levels
(CKO = input; Option 2 = 2,3)
same as Option 29

Option 35: COP Bonding
- = 0: COP420L (28-pin device)
- = 1: COP421L (24-pin device)
- = 2: 28- and 24-pin versions
- = 3: COP422L (20-pin device)
- = 4: 28- and 20-pin versions
- = 5: 24- and 20-pin versions
- = 5: 28-, 24-, and 20-pin versions

Option 36: Internal Initialization Logic
- = 0: normal operation
- = 1: no internal initialization logic

Option Table

The following EPROM option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE = 0 IS: GROUND PIN
OPTION 2 VALUE = _____ IS: CKO OUTPUT
OPTION 3 VALUE = _____ IS: CKI INPUT
OPTION 4 VALUE = _____ IS: RESET INPUT
OPTION 5 VALUE = _____ IS: L_7 DRIVER
OPTION 6 VALUE = _____ IS: L_6 DRIVER
OPTION 7 VALUE = _____ IS: L_5 DRIVER
OPTION 8 VALUE = _____ IS: L_4 DRIVER
OPTION 9 VALUE = _____ IS: IN1 INPUT
OPTION 10 VALUE = _____ IS: IN2 INPUT
OPTION 11 VALUE = _____ IS: VCC PIN
OPTION 12 VALUE = _____ IS: L_3 DRIVER
OPTION 13 VALUE = _____ IS: L_2 DRIVER
OPTION 14 VALUE = _____ IS: L_1 DRIVER
OPTION 15 VALUE = _____ IS: L_0 DRIVER
OPTION 16 VALUE = _____ IS: SI INPUT
OPTION 17 VALUE = _____ IS: SO DRIVER
OPTION 18 VALUE = _____ IS: SK DRIVER

OPTION DATA

OPTION 19 VALUE = _____ IS: IN_0 INPUT
OPTION 20 VALUE = _____ IS: IN_3 INPUT
OPTION 21 VALUE = _____ IS: G_0 I/O PORT
OPTION 22 VALUE = _____ IS: G_1 I/O PORT
OPTION 23 VALUE = _____ IS: G_2 I/O PORT
OPTION 24 VALUE = _____ IS: G_3 I/O PORT
OPTION 25 VALUE = _____ IS: D_3 OUTPUT
OPTION 26 VALUE = _____ IS: D_2 OUTPUT
OPTION 27 VALUE = _____ IS: D_1 OUTPUT
OPTION 28 VALUE = _____ IS: D_0 OUTPUT
OPTION 29 VALUE = _____ IS: L INPUT LEVELS
OPTION 30 VALUE = _____ IS: IN INPUT LEVELS
OPTION 31 VALUE = _____ IS: G INPUT LEVELS
OPTION 32 VALUE = _____ IS: SI INPUT LEVELS
OPTION 33 VALUE = _____ IS: $\overline{\text{RESET}}$ INPUT
OPTION 34 VALUE = _____ IS: CKO INPUT LEVELS
OPTION 35 VALUE = _____ IS: COP BONDING
OPTION 36 VALUE = _____ IS: INTERNAL INITIALIZATION LOGIC

TEST MODE (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the customer-programmed COP420L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

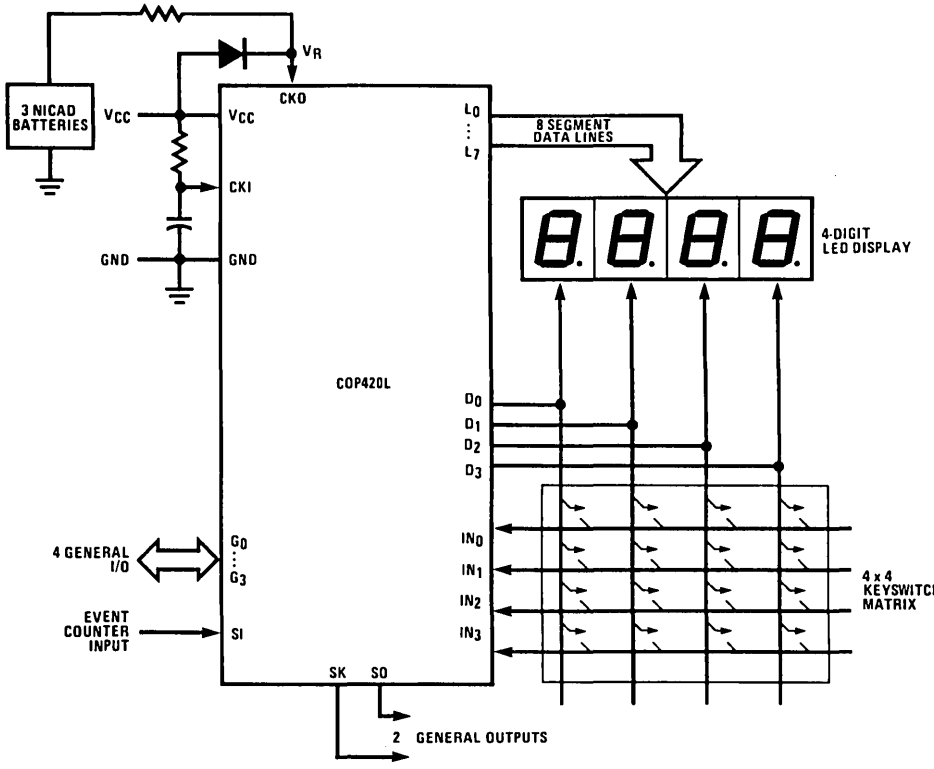
These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATIONS # 1: COP420L General Controller

Figure 9 shows an interconnect diagram for a COP420L used as a general controller. Operation of the system is as follows:

1. The L_7–L_0 outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display.
2. The D_3–D_0 outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN_3–IN_0 inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a V_R RAM power supply pin. RAM data integrity is thereby assured when the main power supply is shut down (see RAM Keep-Alive option description).
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G_3–G_0) are available for use as required by the user's application.

Typical Applications



TL/DD/8825-22

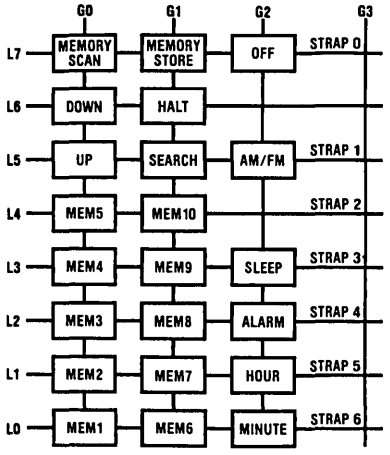
*SO, SI, SK may also be used for Serial I/O

FIGURE 9. COP420L Keyboard/Display Interface

APPLICATION #2:

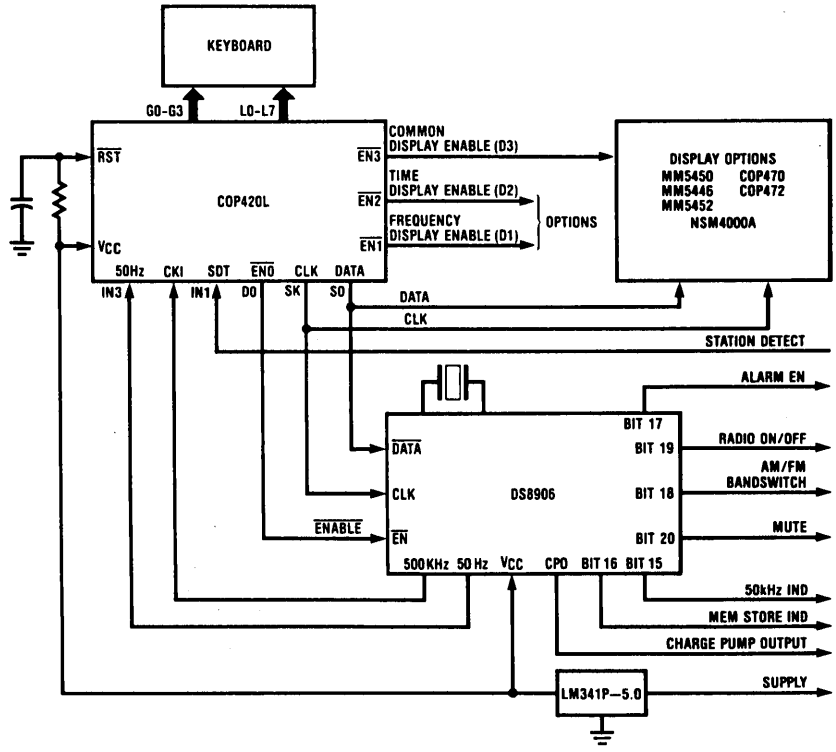
Digitally Tuned Radio Controller and Clock

Keyboard Matrix Configuration



TL/DD/8825-23

Typical Applications (Continued)



TL/DD/8825-24

FIGURE 10. Digital Tuning System Block

Functional Description

LOGIC I/Os

CKI Input: This input accepts an external 500 kHz signal, divides it by eight and outputs the quotient at the CLK output as the system clock.

$\overline{RST}$ Input: Schmitt trigger input to clear device upon initialization.

SDT Input: Interrupt input for station detection. The SDT signal is generated by the radio's station detector and used by the COP420L to determine if there is a valid station on the active frequency. The status of the SDT input is only relevant during station searching mode. A high on SDT will temporarily terminate the search mode for eight seconds.

ALM Input: A high on ALM will activate alarm output via slave device at alarm time. A low on the input will disable alarm function.

DATA Output: Push-pull output providing serial data to external devices.

CLK Output: Push-pull output providing system clock at data transmitting time.

50 Hz Input: A normally high input to accept a 50 Hz external time base for real-time calculation.

MOMENTARY KEYS DESCRIPTION

MEM 1–MEM 10: Each memory represents data of a favorite station in a certain band. Depression of one of these keys will recall the previous stored data and transmit it to the PLL. The PLL will in turn change the radio's receiving frequency as well as the band if necessary. Memory recall keys can also turn on the radio.

UP: This key will manually increment receiving frequency. The first four steps of increment will be for fine tuning a station, after which will be fast slewing meant for manual receive frequency changing.

DOWN: Has the same function as UP key except that frequency is decremented.

MEMORY SCAN: This will start the radio scanning through all ten memories automatically at eight seconds per memory starting from Memory 1. This will also turn on the radio if it was off.

MEMORY STORE: Enables the memory store mode which lasts for three seconds. Depression of any memory key will store the active frequency and band in that memory and disable the store mode. Any function key will also disable the mode to prevent memory data being accidentally destroyed.

HALT: Depression of the HALT key will stop the search and scan functions at current frequency or memory. HALT also turns on the radio during off time and recall frequency display in signal display mode.

SEARCH: Activates station searching in the current band. Search speed is 50 ms per frequency step with wrapping

Functional Description (Continued)

around at end of band. An 8-second stop will take place on reaching a valid station. The HALT key or any function key will terminate the search. Search direction will normally be upwards unless the DOWN key has been depressed prior to the SEARCH key or during the search function in which case search direction will be downwards.

OFF: Turns off the radio or alarm when active.

AM/FM: Radio band switch.

SLEEP: Activates sleep mode, turns on radio on depression and off radio at the end of sleep period. Setting of sleep period is done by depressing the SLEEP and MINUTE key simultaneously.

ALARM: Enables alarm time setting. Depressing the HOUR or MINUTE key and ALARM key simultaneously will set the alarm hour and minute respectively.

HOUR: Sets the hour digits of time-related functions.

MINUTE: Sets the minute digits of time-related functions.

DIODE STRAPS CONNECTIONS

STRAP 0: Controls the on and off of radio. In applications where a toggle type ON/OFF switch is used, momentary OFF key can be omitted; connecting the strap will turn on the radio and vice versa. Must be connected to use momentary OFF key.

STRAP 1, 2: Selects the AM IF options.

STRAP 3: 12/24-hour clock select.

STRAP 4: 3/5 kHz AM step size select.

STRAP 5, 6: FM IF offsets select.

	STRAP 0	STRAP 3	STRAP 4
Connected	Radio ON	12 hour	5 kHz step
Open	Radio OFF	24 hour	3 kHz step

AM/FM IF OPTIONS

AM	STRAP 1	STRAP 2
455 kHz	X	X
460 kHz	X	✓
450 kHz	✓	X
260 kHz	✓	✓
FM	**STRAP 5**	**STRAP 6**
10.7 MHz	X	X
10.75 MHz	X	✓
10.65 MHz	✓	X
10.8 MHz	✓	✓

X = No connection.

✓ = Diode inserted.

INDIRECT FEATURES AND OPTIONS

As indicated in *Figure 10,* there are a few options and indirect features provided via the help of a slave device, namely the Phase Lock Loop, DS8906N.

DISPLAY OPTIONS

As mentioned above, the COP420L-HSB is MICROWIRE compatible. Internal circuitry enables it to directly interface with all of National's serial input MICROWIRE compatible display drivers whether they are of a direct drive or multiplex drive format. On *Figure 10* is a list of drivers available for the system. EN1 and EN2 are optional enable outputs meant for a dual display system in which EN3 will not be used. By dual display, it means that one display will be constantly showing time information and the other showing frequency information. Whereas in conventional single display systems, the display shows both time and frequency information in a time-sharing method. The National system provides a time-prioritized display-sharing method. That is, whenever a tuning function is completed, the frequency information will stay on the display for eight seconds then time display will take over. This is achieved by using EN3 for the driver's enable logic.

CONTROL OUTPUTS

Six open collector outputs controlled by the COP420L are provided from DS8906N, the phase lock loop for controlling radio switching circuits.

Radio ON/OFF: A high from this output indicates that the radio should be switched on and vice versa.

AM/FM: Output for controlling the AM/FM bandswitch. A high level output indicates FM and a low indicates the AM band.

MUTE: For muting the audio output when performing any frequency related function. The output will go high prior to the frequency change except when doing fine tuning.

ALARM ENABLE: Active high output for turning on the alarm circuit at alarm time.

50 kHz IND: For driving the 50 kHz indicator in FM band or the LSB in a 5-digit display. Output is active high.

MEM STORE IND: For driving the memory store mode indicator. Output is active high.

TYPICAL IMPLEMENTATION ALTERNATIVES

A full keyboard or any portion of it can be implemented with various applications for features/functions vs. cost/size.

Figure 11 shows two keyboard configurations with 22-key and 11-key keyboards for a desk-top/tuner system or auto-radio system, respectively.

Functional Description (Continued)

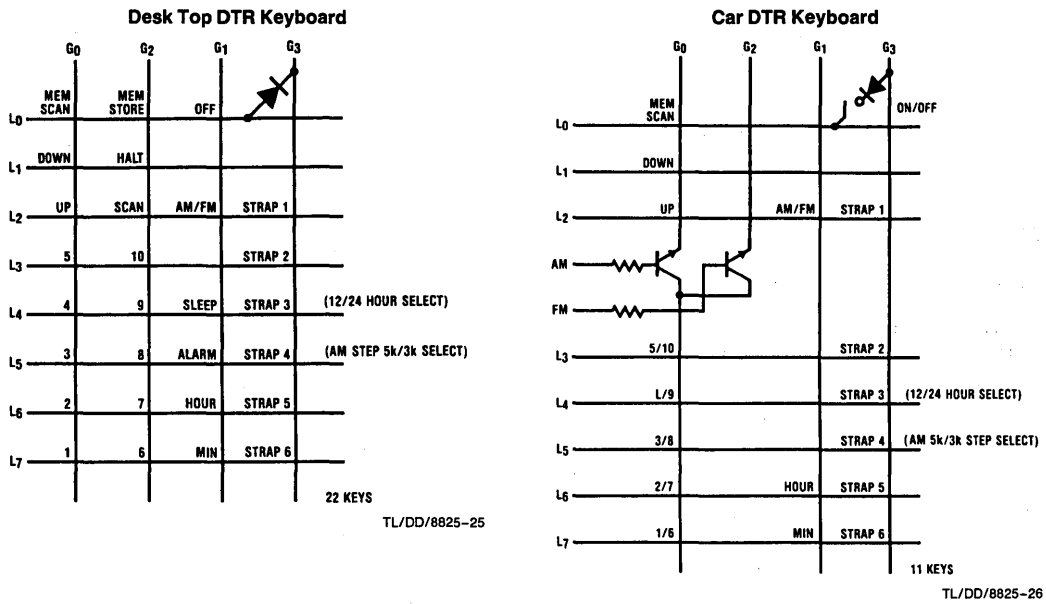


FIGURE 11

National Semiconductor

COP424C, COP425C, COP426C, COP324C, COP325C, COP326C and COP444C, COP445C, COP344C, COP345C Single-Chip 1k and 2k CMOS Microcontrollers

General Description

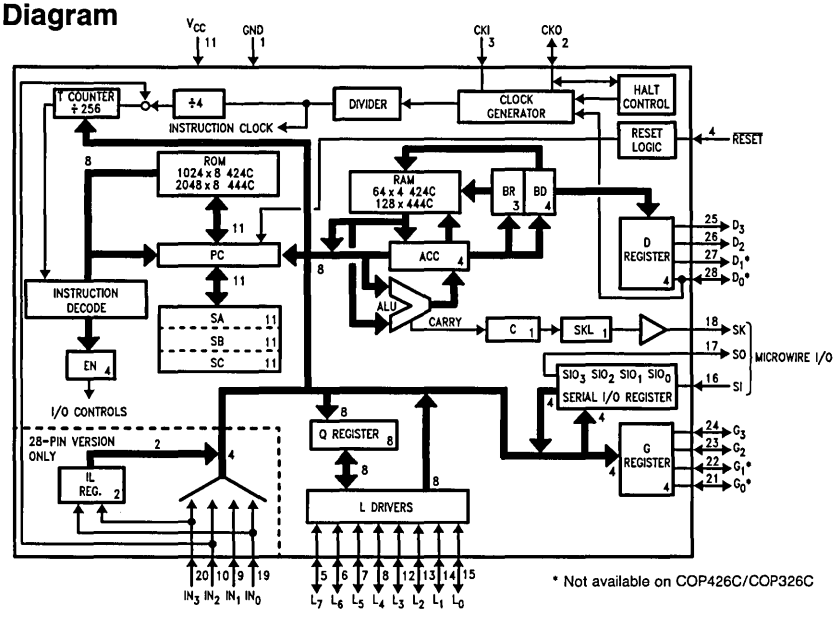
The COP424C, COP425C, COP426C, COP444C and COP445C fully static, Single-Chip CMOS Microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon gate microCMOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP424C and COP444C are 28 pin chips. The COP425C and COP445C are 24-pin versions (4 inputs removed) and COP426C is 20-pin version with 15 I/O lines. Standard test procedures and reliable high-density techniques provide the medium to large volume customers with a customized microcontroller at a low end-product cost. These microcontrollers are appropriate choices in many demanding control environments especially those with human interface.

The COP424C is an improved product which replaces the COP420C.

Features

- Lowest power dissipation (50 μW typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4 μs instruction time, plus software selectable clocks
- 2k x 8 ROM, 128 x 4 RAM (COP444C/COP445C)
- 1k x 8 ROM, 64 x 4 RAM (COP424C/COP425C/COP426C)
- 23 I/O lines (COP444C and COP424C)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (2.4V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS output compatible
- Microbus™ compatible
- Software/hardware compatible with COP400 family
- Extended temperature range devices COP324C/COP325C/COP326C and COP344C/COP345C (−40°C to +85°C)
- Military devices (−55°C to +125°C) to be available

Block Diagram



* Not available on COP426C/COP326C

TL/DD/5259–1

FIGURE 1

COP424C/COP425C/COP426C and COP444C/COP445C

Absolute Maximum Ratings

Supply Voltage (V_{CC})	6V
Voltage at any Pin	−0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 0°C≤T_A≤70°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple (Note 5)	Peak to Peak		0.1 V_{CC}	V
Supply Current (Note 1)	V_{CC}=2.4V, tc=64 μs		120	μA
	V_{CC}=5.0V, tc=16 μs		700	μA
	V_{CC}=5.0V, tc=4 μs		3000	μA
	(tc is instruction cycle time)			
HALT Mode Current (Note 2)	V_{CC}=5.0V, F_{IN}=0 kHz		40	μA
	V_{CC}=2.4V, F_{IN}=0 kHz		12	μA
Input Voltage Levels				
RESET, CKI, D_0 (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	V_{CC}=4.5V, V_{IN}=0	30	330	μA
Hi-Z Input Leakage		−1	+1	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels	Standard Outputs			
LSTTL Operation	V_{CC}=5.0V±10%			
Logic High	I_{OH}=−100 μA	2.7		V
Logic Low	I_{OL}=400 μA		0.4	V
CMOS Operation				
Logic High	I_{OH}=−10 μA	V_{CC}−0.2		V
Logic Low	I_{OL}=10 μA		0.2	V
Output Current Levels (except CKO)				
Sink (Note 6)	V_{CC}=4.5V, V_{OUT}=V_{CC}	1.2		mA
	V_{CC}=2.4V, V_{OUT}=V_{CC}	0.2		mA
Source (Standard Option)	V_{CC}=4.5V, V_{OUT}=0V	−0.5		mA
	V_{CC}=2.4V, V_{OUT}=0V	−0.1		mA
Source (Low Current Option)	V_{CC}=4.5V, V_{OUT}=0V	−30	−330	μA
	V_{CC}=2.4V, V_{OUT}=0V	−6	−80	μA
CKO Current Levels (As Clock Out)				
Sink ÷4	V_{CC}=4.5V, CKI=V_{CC}, V_{OUT}=V_{CC}	0.3		mA
÷8		0.6		mA
÷16		1.2		mA
Source ÷4	V_{CC}=4.5V, CKI=0V, V_{OUT}=0V	−0.3		mA
÷8		−0.6		mA
÷16		−1.2		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	V_{CC}=4.5V, V_{IN}=0.2V_{CC}		0.7	mA
To Halt	V_{CC}=4.5V, V_{IN}=0.7V_{CC}		1.6	mA
TRI-STATE or Open Drain Leakage Current		−2.5	+2.5	μA

COP324C/COP325C/COP326C and COP344C/COP345C

Absolute Maximum Ratings

Supply Voltage	6V
Voltage at any Pin	−0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		3.0	5.5	V
Power Supply Ripple (Note 5)	Peak to Peak		0.1 V_{CC}	V
Supply Current (Note 1)	V_{CC} = 3.0V, tc = 64 μs		180	μA
	V_{CC} = 5.0V, tc = 16 μs		800	μA
	V_{CC} = 5.0V, tc = 4 μs (tc is instruction cycle time)		3600	μA
HALT Mode Current (Note 2)	V_{CC} = 5.0V, F_{IN} = 0 kHz		60	μA
	V_{CC} = 3.0V, F_{IN} = 0 kHz		30	μA
Input Voltage Levels				
RESET, CKI, D_O (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	V_{CC} = 4.5V, V_{IN} = 0	30	440	μA
Hi-Z Input Leakage		−2	+2	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels	Standard Outputs			
LSTTL Operation	V_{CC} = 5.0V ±10%			
Logic High	I_{OH} = −100 μA	2.7		V
Logic Low	I_{OL} = 400 μA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC} − 0.2		V
Logic Low	I_{OL} = 10 μA		0.2	V
Output Current Levels (except CKO)				
Sink (Note 6)	V_{CC} = 4.5V, V_{OUT} = V_{CC}	1.2		mA
	V_{CC} = 3.0V, V_{OUT} = V_{CC}	0.2		mA
Source (Standard Option)	V_{CC} = 4.5V, V_{OUT} = 0V	−0.5		mA
	V_{CC} = 3.0V, V_{OUT} = 0V	−0.1		mA
Source (Low Current Option)	V_{CC} = 4.5V, V_{OUT} = 0V	−30	−440	μA
	V_{CC} = 3.0V, V_{OUT} = 0V	−8	−200	μA
CKO Current Levels (As Clock Out)				
Sink ÷4	V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC}	0.3		mA
÷8		0.6		mA
÷16		1.2		mA
Source ÷4	V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V	−0.3		mA
÷8		−0.6		mA
÷16		−1.2		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	V_{CC} = 4.5V, V_{IN} = 0.2V_{CC}		0.9	mA
To Halt	V_{CC} = 4.5V, V_{IN} = 0.7V_{CC}		2.1	mA
TRI-STATE or Open Drain Leakage Current		−5	+5	μA

COP424C/COP425C/COP426C and COP444C/COP445C

AC Electrical Characteristics $0°C \le T_A \le 70°C$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (tc)	$V_{CC} \ge 4.5V$	4	DC	μs
	$4.5V > V_{CC} \ge 2.4V$	16	DC	μs
Operating CKI Frequency ÷4 mode	$V_{CC} \ge 4.5V$	DC	1.0	MHz
÷8 mode	$V_{CC} \ge 4.5V$	DC	2.0	MHz
÷16 mode	$V_{CC} \ge 4.5V$	DC	4.0	MHz
÷4 mode	$4.5V > V_{CC} \ge 2.4V$	DC	250	kHz
÷8 mode	$4.5V > V_{CC} \ge 2.4V$	DC	500	kHz
÷16 mode	$4.5V > V_{CC} \ge 2.4V$	DC	1.0	MHz
Duty Cycle (Note 4)	$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 4$ MHz External Clock		60	ns
Fall Time (Note 4)	$f_1 = 4$ MHz External Clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)	$R = 30k$, $V_{CC} = 5V$, $C = 82$ pF (÷4 Mode)	8	16	μs
Inputs: (See *Figure 3*)				
t_{SETUP} G Inputs	$V_{CC} \ge 4.5V$	tc/4+.7		μs
t_{SETUP} SI Input	$V_{CC} \ge 4.5V$	0.3		μs
t_{SETUP} All Others	$V_{CC} \ge 4.5V$	1.7		μs
t_{HOLD}	$V_{CC} \ge 4.5V$	0.25		μs
	$4.5V > V_{CC} \ge 2.4V$	1.0		μs
Output Propagation Delay	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5k$			
t_{PD1}, t_{PD0}	$V_{CC} \ge 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$4.5V > V_{CC} \ge 2.4V$		4.0	μs
Microbus Timing	$CL = 50$ pF, $V_{CC} = 5V \pm 5\%$			
Read Operation (*Figure 4*)				
Chip Select Stable before $\overline{RD}$ $-t_{CSR}$		65		ns
Chip Select Hold Time for $\overline{RD}$ $-t_{RCS}$		20		ns
$\overline{RD}$ Pulse Width $-t_{RR}$		400		ns
Data Delay from $\overline{RD}$ $-t_{RD}$			375	ns
$\overline{RD}$ to Data Floating $-t_{DF}$ (Note 4)			250	ns
Write Operation (*Figure 5*)				
Chip Select Stable before $\overline{WR}$ $-t_{CSW}$		65		ns
Chip Select Hold Time for $\overline{WR}$ $-t_{WCS}$		20		ns
$\overline{WR}$ Pulse Width $-t_{WW}$		400		ns
Data Set-Up Time for $\overline{WR}$ $-t_{DW}$		320		ns
Data Hold Time for $\overline{WR}$ $-t_{WD}$		100		ns
INTR Transition Time from $\overline{WR}$ $-t_{WI}$			700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 17.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}, L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than $0.2V_{CC}$ when part is running in order to prevent entering test mode.

COP324C/COP325C/COP326C and COP344C/COP345C

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (tc)	$V_{CC} \ge 4.5V$	4	DC	μs
	$4.5V > V_{CC} \ge 3.0V$	16	DC	μs
Operating CKI Frequency ÷4 mode	$V_{CC} \ge 4.5V$	DC	1.0	MHz
÷8 mode	$V_{CC} \ge 4.5V$	DC	2.0	MHz
÷16 mode	$V_{CC} \ge 4.5V$	DC	4.0	MHz
÷4 mode	$4.5V > V_{CC} \ge 3.0V$	DC	250	kHz
÷8 mode	$4.5V > V_{CC} \ge 3.0V$	DC	500	kHz
÷16 mode	$4.5V > V_{CC} \ge 3.0V$	DC	1.0	MHz
Duty Cycle (Note 4)	$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 4$ MHz external clock		60	ns
Fall Time (Note 4)	$f_1 = 4$ MHz external clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)	R = 30k, V_{CC} = 5V C = 82 pF (÷4 Mode)	8	16	μs
Inputs: (See *Figure 3*)				
t_{SETUP} G Inputs	$V_{CC} \ge 4.5V$	tc/4+.7		μs
t_{SETUP} SI Inputs	$V_{CC} \ge 4.5V$	0.3		μs
t_{SETUP} All Others	$V_{CC} \ge 4.5V$	1.7		μs
t_{HOLD}	$V_{CC} \ge 4.5V$	0.25		μs
	$4.5V > V_{CC} \ge 3.0V$	1.0		μs
Output Propagation Delay	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5k$			
t_{PD1}, t_{PD0}	$V_{CC} \ge 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$4.5V > V_{CC} \ge 3.0V$		4.0	μs
Microbus Timing	$C_L = 50$ pF, $V_{CC} = 5V \pm 5\%$			
Read Operation (*Figure 4*)				
Chip Select Stable before $\overline{RD}$ – t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$ – t_{RCS}		20		ns
$\overline{RD}$ Pulse Width – t_{RR}		400		ns
Data Delay from $\overline{RD}$ – t_{RD}			375	ns
$\overline{RD}$ to Data Floating – t_{DF} (Note 4)			250	ns
Write Operation (*Figure 5*)				
Chip Select Stable before $\overline{WR}$ – t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$ – t_{WCS}		20		ns
$\overline{WR}$ Pulse Width – t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$ – t_{DW}		320		ns
Data Hold Time for $\overline{WR}$ – t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$ – t_{WI}			700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 17.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}, L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

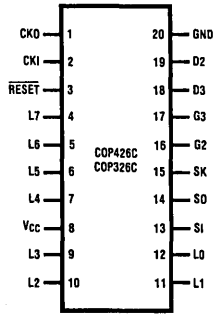
Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than $0.2V_{CC}$ when part is running in order to prevent entering test mode.

Connection Diagrams

DIP and S.O. Wide



TL/DD/5259-16

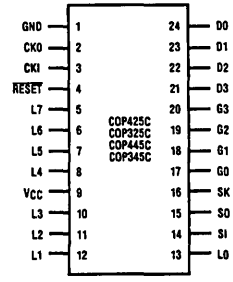
Top View

Order Number COP326C-XXX/D or COP426C-XXX/D
See NS Hermetic Package D20A

Order Number COP326C-XXX/N or COP426C-XXX/N
See NS Molded Package N20A

Order Number COP326C-XXX/WM or COP426C-XXX/WM
See NS Surface Mount Package M20B

DIP and S.O. Wide



TL/DD/5259-2

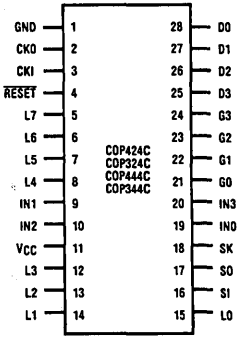
Top View

Order Number COP325C-XXX/D or COP425C-XXX/D
See NS Hermetic Package D24C

Order Number COP325C-XXX/N or COP425C-XXX/N
See NS Molded Package N24A

Order Number COP325C-XXX/WM or COP425C-XXX/WM
See NS Surface Mount Package M24B

Dual-In-Line Package



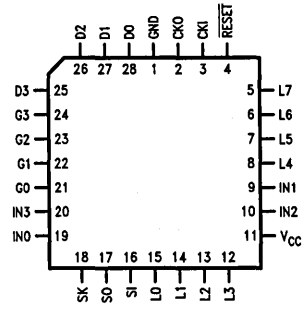
TL/DD/5259-3

Top View

Order Number COP324C-XXX/D or COP424C-XXX/D
See NS Hermetic Package D28C

Order Number COP324C-XXX/N or COP424C-XXX/N
See NS Molded Package N28B

PLCC



TL/DD/5259-18

Order Number COP324C-XXX/V or COP424C-XXX/V
See NS PLCC Package V28A

FIGURE 2

Pin	Description
L7–L0	8-bit bidirectional port with TRI-STATE
G3–G0	4-bit bidirectional I/O port
D3–D0	4-bit output port
IN3–IN0	4-bit input port (28-pin package only)
SI	Serial input or counter input
SO	Serial or general purpose output

Pin	Description
SK	Logic controlled clock output
CKI	Chip oscillator input
CKO	Oscillator output, HALT I/O port or general purpose input
$\overline{\text{RESET}}$	Reset input
V_{CC}	Most positive power supply
GND	Ground

Functional Description

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

For ease of reading only the COP424C/425C/COP426C/444C/445C are referenced; however, all such references apply equally to COP324C/325C/COP326C/344C/345C.

PROGRAM MEMORY

Program Memory consists of ROM, 1024 bytes for the COP424C/425C/426C and 2048 bytes for the COP444C/445C. These bytes of ROM may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by a 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

Data memory consists of a 512-bit RAM for the COP444C/445C, organized as 8 data registers of 16 × 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register.

Data memory consists of a 256-bit RAM for the COP424C/425C/426C, organized as 4 data registers of 16 × 4-bits digits. The B register is 6 bits long. Upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions.

The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, to input 4 bits of L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes depending on a mask-programmable option: as a timer or as an external event counter. When the T counter overflows, an overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 10a*.

Four general-purpose inputs, IN3-IN0, are provided. IN1, IN2 and IN3 may be selected, by a mask-programmable option as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in Microbus application.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. In the dual clock mode, D0 latch controls the clock selection (see dual oscillator below).

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G0 may be mask-programmed as an output for Microbus applications.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the Microbus option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M. As explained above, the Microbus option allows L I/O port data to be latched into the Q register.

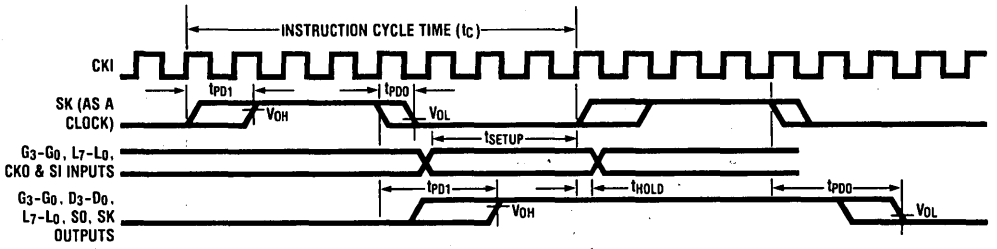
Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

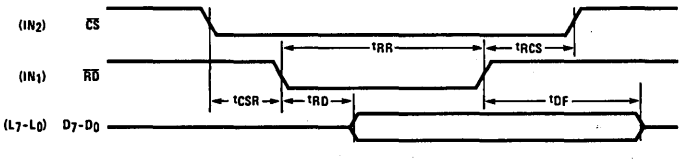
EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.



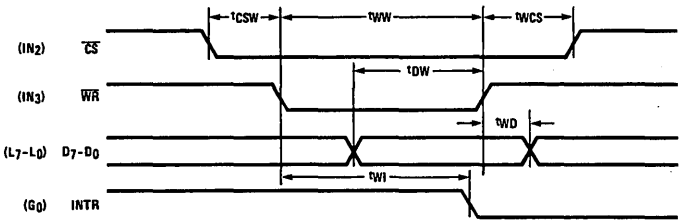
TL/DD/5259-4

FIGURE 3. Input/Output Timing Diagrams (divide by 8 mode)



TL/DD/5259-5

FIGURE 4. Microbus Read Operation Timing



TL/DD/5259-6

FIGURE 5. Microbus Write Operation Timing

Functional Description (Continued)

3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.

b. An interrupt will be recognized only on the following conditions:
   1. EN1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

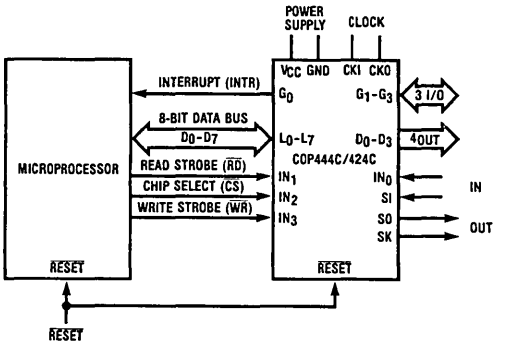
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

MICROBUS INTERFACE

The COP444C/424C has an option which allows it to be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN1, IN2 and IN3 general purpose inputs become Microbus compatible read-strobe, chip-select, and write-strobe lines, respectively. IN1 becomes $\overline{RD}$ — a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the uP. IN2 becomes $\overline{CS}$ — a logic "0" on this line selects the COP444C/424C as the uP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN3 becomes $\overline{WR}$ — a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP444C/424C. G0 becomes INTR a "ready" output, reset by a write pulse from the uP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP444C/424C.

This option has been designed for compatibility with National's Microbus — a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See Microbus National Publication.) The functioning and timing relationships between the signal lines affected by this option are as specified for the Microbus interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection of the COP444C/424C to the Microbus is shown in *Figure 6*.



TL/DD/5259-7

FIGURE 6. Microbus Option Interconnect

TABLE I. Enable Register Modes — Bits EN0 and EN3

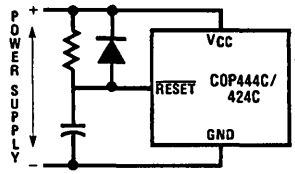
EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1,SK=clock If SKL=0,SK=0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL=1,SK=clock If SKL=0,SK=0
1	0	Binary Counter	Input to Counter	0	SK=SKL
1	1	Binary Counter	Input to Counter	1	SK=SKL

Functional Description (Continued)

INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 7* must be connected to the $\overline{\text{RESET}}$ pin (the conditions in *Figure 7* must be met). The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

Note: If CKI clock is less than 32 kHz, the internal reset logic (option #29 = 1) MUST be disabled and the external RC circuit must be used.



RC≥5X POWER SUPPLY RISE TIME
AND RC≥100X CKI PERIOD.

TL/DD/5259–8

FIGURE 7. Power-Up Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).

TIMER

There are two modes selected by mask option:

a. Time-base counter. In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset.

For example, using a 4 MHz crystal with a divide-by-16 option, the instruction cycle frequency of 250 kHz increments the 10-bit timer every 4 μs. By presetting the counter and detecting overflow, accurate timeouts between 16 μs (4 counts) and 4.096 ms (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

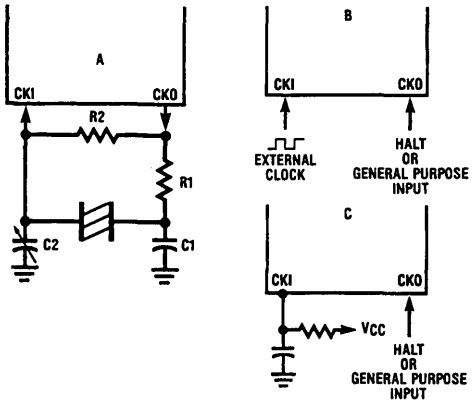
b. External event counter. In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

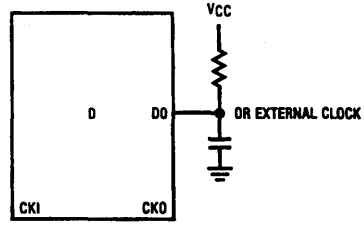
Note: The IT instruction is not allowed in this mode.

HALT MODE

The COP444C/445C/424C/425C/426C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by the HALT instruction or by forcing CKO high when it is mask-programmed as an HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The chip may be awakened by one of two different methods:

- Continue function: by forcing CKO low, if it mask-programmed as an HALT I/O port, the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization).





TL/DD/5259–9

Crystal or Resonator

Crystal Value	Component Values			
	R1	R2	C1(pF)	C2(pF)
32 kHz	220k	20M	30	6–36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6–36
4.0 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R	C	Cycle Time	V_{CC}
15k	82 pF	4–9 μs	≥4.5V
30k	82 pF	8–16 μs	≥4.5V
60k	100 pF	16–32 μs	2.4–4.5V

Note: 15k≤R≤150k
50 pF≤C≤150 pF

FIGURE 8. Oscillator Component Values

Functional Description (Continued)

The HALT mode is the minimum power dissipation state.

Note: If the user has selected dual-clock with D0 as external oscillator (option 30=2) AND the COP444C/424C is running with the D0 clock, the HALT mode — either hardware or software — will NOT be entered. Thus, the user should switch to the CKI clock to HALT. Alternatively, the user may stop the D0 clock to minimize power.

CKO PIN OPTIONS

a. Two-pin oscillator — (Crystal). See *Figure 9A*.

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. The HALT mode may be entered by program control (HALT instruction) which forces CKO high, thus inhibiting the crystal network. The circuit can be awakened only by forcing the $\overline{\text{RESET}}$ pin to a logic "0" (restart).

b. One-pin oscillator — (RC or external). See *Figure 9B*.

If a one-pin oscillator system is chosen, two options are available for CKO:

- CKO can be selected as the HALT I/O port. In that case, it is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and CKO output will stay high to keep the chip stopped if the external driver returns to high impedance state. By forcing a low level to CKO, the chip will continue and CKO will stay low.
- As another option, CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction.

OSCILLATOR OPTIONS

There are four basic clock oscillator configurations available as shown by *Figure 8*.

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency optionally divided by 4, 8 or 16.

b. External Oscillator. The external frequency is optionally divided by 4, 8 or 16 to give the instruction cycle time. CKO is the HALT I/O port or a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port or a general purpose input.

d. Dual oscillator. By selecting the dual clock option, pin D0 is now a single pin oscillator input. Two configurations are available: RC controlled Schmitt trigger oscillator or external oscillator.

The user may software select between the D0 oscillator (in that case, the instruction cycle time equals the D0 oscillation frequency divided by 4) by setting the D0 latch high or the CKI (CKO) oscillator by resetting D0 latch low. Note that even in dual clock mode, the counter, if mask-programmed as a time-base counter, is always connected to the CKI oscillator.

For example, the user may connect up to a 1 MHz RC circuit to D0 for faster processing and a 32 kHz watch crystal to CKI and CKO for minimum current drain and time keeping.

Note: CTMA instruction is not allowed when chip is running from D0 clock.

Figures 10A and *10B* show the clock and timer diagrams with and without Dual clock.

COP445C AND COP425C 24-PIN PACKAGE OPTION

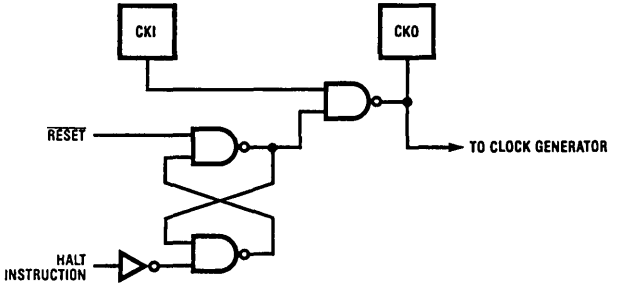
If the COP444C/424C is bonded in a 24-pin package, it becomes the COP445C/425C, illustrated in *Figure 2*, Connection diagrams. Note that the COP445C/425C does not contain the four general purpose IN inputs (IN3–IN0). Use of this option precludes, of course, use of the IN options, interrupt feature, external event counter feature, and the Microbus option which uses IN1–IN3. All other options are available for the COP445C/425C.

Note: If user selects the 24-pin package, options 9, 10, 19 and 20 must be selected as a "0" (load to V_{CC} on the IN inputs). See option list.

COP426C 20-PIN PACKAGE OPTION

If the COP425C is bonded as 20-pin device it becomes the COP426C. Note that the COP426C contains all the COP425C pins except D_0, D_1, G_0, and G_1.

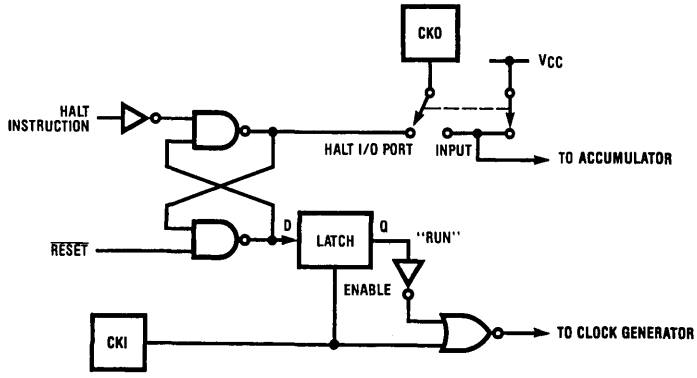
Block Diagram (Continued)



TL/DD/5259–10

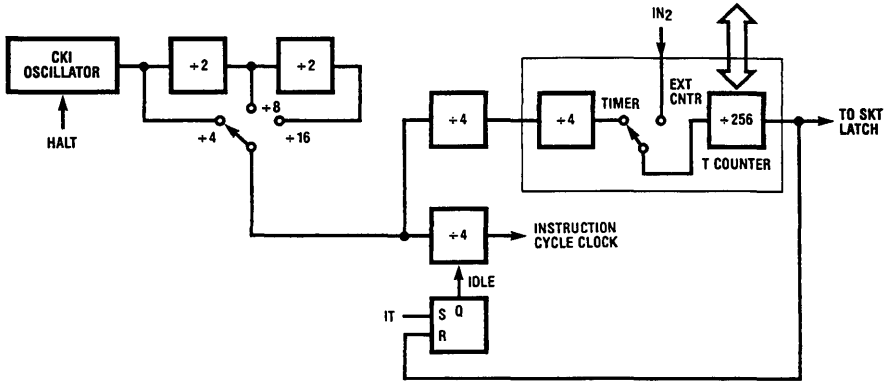
FIGURE 9A. Halt Mode — Two-Pin Oscillator

Block Diagram (Continued)



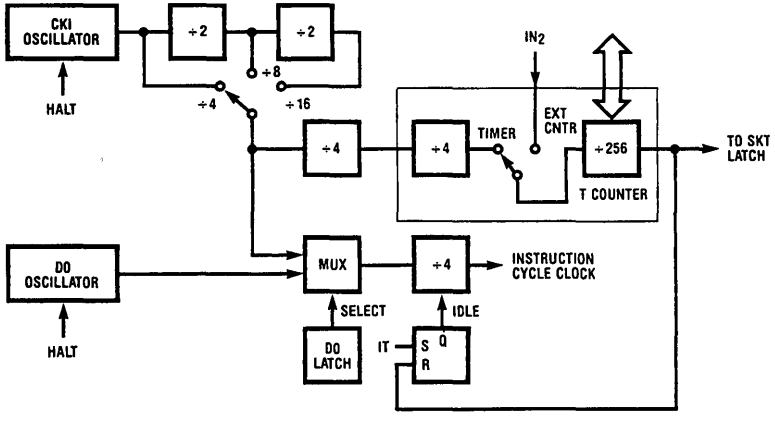
TL/DD/5259–11

FIGURE 9B. Halt Mode — One-Pin Oscillator



TL/DD/5259–12

FIGURE 10A. Clock and Timer without Dual-Clock



TL/DD/5259–13

FIGURE 10B. Clock and Timer with Dual-Clock

Instruction Set

Table II is a symbol table providing internal architecture, instruction operan and operation symbols used in the instruction set table.

TABLE II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit accumulator
B	7-bit RAM address register (6-bit for COP424C)
Br	Upper 3 bits of B (register address) (2-bit for COP424C)
Bd	Lower 4 bits of B (digit address)
C	1-bit carry register
D	4-bit data output port
EN	4-bit enable register
G	4-bit general purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA,SB,SC	11-bit 3-level subroutine stack
SIO	4-bit shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

Instruction Operand Symbols	
d	4-bit operand field, 0–15 binary (RAM digit select)
r	3(2)-bit operand field, 0–7(3) binary (RAM register select)
a	11-bit operand field, 0–2047 (1023)
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x

Operational Symbols	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	One's complement of A
⊕	Exclusive-or
:	Range of values

TABLE III. COP444C/445C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A+RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	$A+10_{10}$ → A	None	Add Ten to A
AISC	y	5–	|0101| y |	A+y → A	Carry	Add Immediate. Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$+RAM(B)+C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A⊕RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM ($PC_{10:8}$ A,M) → $PC_{7:0}$	None	Jump Indirect (Notes 1, 3)
JMP	a	6–	0110 | 0 | $a_{10:8}$	a → PC	None	Jump
		––	$a_{7:0}$			
JP	a	––	1 | $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 4)
			or			
		––	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	––	10 | $a_{5:0}$	PC+1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6–	0110 | 1 | $a_{10:8}$	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
		––	$a_{7:0}$			
RET		48	0100 | 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33	0011 | 0011		None	HALT Processor
		38	0011 | 1000			
IT		33	0011 | 0011			IDLE till Timer
		39	0011 | 1001		None	Overflows then Continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33	0011 | 0011	A → $T_{7:4}$		
		3F	0011 | 1111	RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CTMA		33	0011 | 0011	$T_{7:4}$ → RAM(B)		
		2F	0010 | 1111	$T_{3:0}$ → A	None	Copy T to RAM, A (Note 9)
CAMQ		33	0011 | 0011	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 | 1100	RAM(B) → $Q_{3:0}$		
CQMA		33	0011 | 0011	$Q_{7:4}$ → RAM(B)	None	Copy Q to RAM, A
		2C	0010 | 1100	$Q_{3:0}$ → A		
LD	r	–5	00 | r | 0101 (r=0:3)	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23	0010 | 0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		––	0 | r | d			
LQID		BF	1011 | 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7–	|0111| y |	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate 1 and Increment Bd
X	r	–6	|00| r |0110| (r=0:3)	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 ––	|0010|0011| |1| r | d |	RAM(r,d) ⟷ A	None	Exchange A with RAM Pointed to Directly by r,d
XDS	r	–7	|00 |r|0111| (r=0:3)	RAM(B) ⟷ A Bd–1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	–4	|00 | r |0100| (r=0:3)	RAM(B) ⟷ A Bd+1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	––	|00|r|(d–1)| (r=0:3: d=0,9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33 ––	|0011|0011| |1| r | d | (any r, any d)			
LEI	y	33 6–	|0011|0011| |0110| y |	y → EN	None	Load EN Immediate (Note 7)
XABR		12	|0001|0010|	A ⟷ Br	None	Exchange A with Br (Note 8)
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C="1"	Skip if C is True
SKE		21	|0010|0001|		A=RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	$G_0=0$	
	1	11	|0001|0001|		$G_1=0$	
	2	03	|0000|0011|		$G_2=0$	
	3	13	|0001|0011|		$G_3=0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1=0$	
	2	03	|0000|0011|		$RAM(B)_2=0$	
	3	13	|0001|0011|		$RAM(B)_3=0$	
SKT		41	|0100|0001|		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 | 0011	G → A	None	Input G Ports to A
		2A	0010 | 1010			
ININ		33	0011 | 0011	IN → A	None	Input IN Inputs to A (Note 2)
		28	0010 | 1000			
INIL		33	0011 | 0011	IL_3, CKO,"0", IL_0 → A	None	Input IL Latches to A (Note 3)
		29	0010 | 1001			
INL		33	0011 | 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM,A
		2E	0010 | 1110	$L_{3:0}$ → A		
OBD		33	0011 | 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 | 1110			
OGI	y	33	0011 | 0011	y → G	None	Output to G Ports Immediate
		5–	0101 | y			
OMG		33	0011 | 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin packages since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B(Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: For 2K ROM devices, A ⟷ Br (0 → A3). For 1K ROM devices, A ⟷ Br (0,0 → A3, A2).

Note 9: Do not use CTMA instruction when dual-clock option is selected and part is running from D_0 clocks.

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10:PC8,A,M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC+1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC(7:4), RAM(B) → PC(3:0), leaving PC(10), PC(9) and PC(8) unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10:8,A,M. PC10,PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT   ; load T counter
SKT    ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is mask-programmed as an external event counter (option #31 = 1).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKO and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A0 is input into A1. IL latches are cleared on reset. IL latches are not available on the COP445C/425C, and COP426C.

INSTRUCTION SET NOTES

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.

c. The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Note: The COP424C/425C/426C needs only 10 bits to address its ROM. Therefore, the eleventh bit (P10) is ignored.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal or resonator generated clock input will draw additional current. An R/C oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate operating current drain.

$I_{CO} = I_Q + V \times 40 \times Fi + V \times 1400 \times Fi/Dv$

where I_{CO} = chip operating current drain in microamps
I_Q quiescent leakage current (from curve)
Fi CKI frequency in MegaHertz
V chip V_{CC} in volts
Dv divide by option selected

For example at 5 volts V_{CC} and 400 kHz (divide by 4)

$I_{CO} = 20 + 5 \times 40 \times 0.4 + 5 \times 1400 \times 0.4/4$

$I_{CO} = 20 + 80 + 700 = 800\ \mu A$

At 2.4 volts V_{CC} and 30 kHz (divide by 4)

$I_{CO} = 6 + 2.4 \times 40 \times 0.03 + 2.4 \times 1400 \times 0.03/4$

$I_{CO} = 6 + 2.88 + 25.2 = 34.08\ \mu A$

Power Dissipation (Continued)

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$Ici = I_Q + V \times 40 \times Fi$$

For example, at 5 volts V_{CC} and 400 kHz

$$Ici = 20 + 5 \times 40 \times 0.4 = 100\ \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$Ita = I_{CO} \times \frac{To}{To+Ti} + Ici \times \frac{Ti}{To+Ti}$$

where: Ita = total average current
I_{CO} = operating current
Ici = idle current
To = operating time
Ti = idle time

I/O OPTIONS

Outputs have the following optional configurations, illustrated in *Figure 11*:

a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

b. Low Current — This is the same configuration as a. above except that the sourcing current is much less.

c. Open Drain — An N-channel device to ground only, allowing external pull-up as required by the user's application.

d. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control.

e. Low-Current TRI-STATE L Output — This is the same as d. above except that the sourcing current is much less.

f. Open-Drain TRI-STATE L Output — This has the N-channel device to ground only.

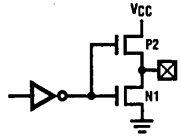
All inputs have the following options:

g. Input with on chip load device to V_{CC}.

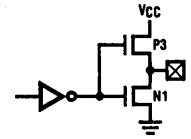
h. Hi-Z input which must be driven by the users logic.

When using either the G or L I/O ports as inputs, a pull-up device is necessary. This can be an external device or the following alternative is available: Select the low-current output option. Now, by setting the output registers to a logic "1" level, the P-channel devices will act as the pull-up load. Note that when using the L ports in this fashion the Q registers must be set to a logic "1" level and the L drivers MUST BE ENABLED by an LEI instruction (see description above).

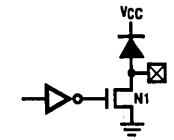
All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 12* for each of these devices to allow the designer to effectively use these I/O configurations.



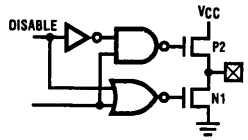
a. Standard Push-Pull Output



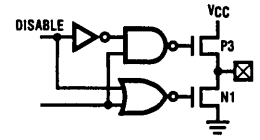
b. Low Current Push-Pull Output



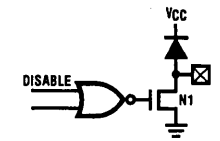
c. Open-Drain Output



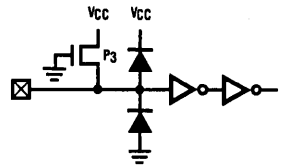
d. Standard TRI-STATE "L" Output



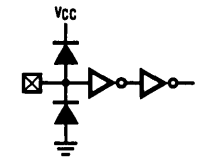
e. Low Current TRI-STATE "L" Output



f. Open Drain TRI-STATE "L" Output



g. Input with Load

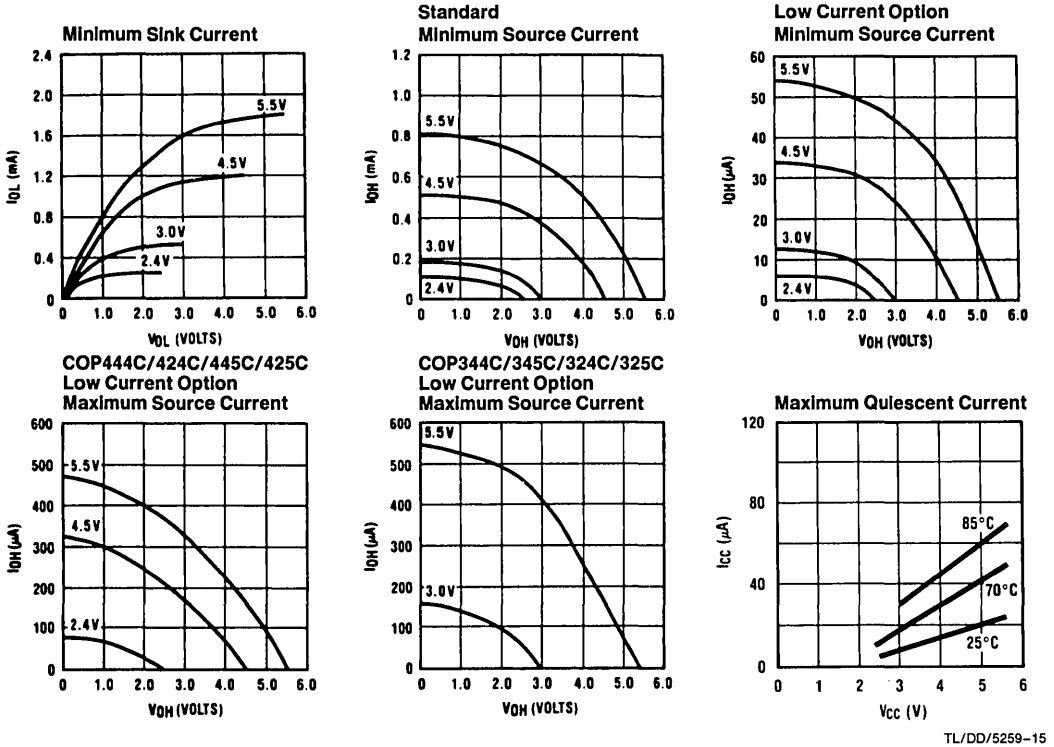


h. HI-Z Input

TL/DD/5259–14

FIGURE 11. Input/Output Configurations

Power Dissipation (Continued)



TL/DD/5259–15

FIGURE 12. Input/Output Characteristics

Option List

The COP444C/445C/424C/425C/COP426C mask-programmable options are assigned numbers which correspond with the COP444C/424C pins.

The following is a list of options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

PLEASE FILL OUT THE OPTION TABLE on the next page. Xerox the option data and send it in with your disk or EPROM.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Pin
- = 0: clock generator output to crystal/resonator
- = 1: HALT I/O port
- = 2: general purpose input with load device to V_{CC}
- = 3: general purpose input, high-Z

Option 3: CKI input
- = 0: Crystal controlled oscillator input divide by 4
- = 1: Crystal controlled oscillator input divide by 8
- = 2: Crystal controlled oscillator input divide by 16
- = 4: Single-pin RC controlled oscillator (divide by 4)
- = 5: External oscillator input divide by 4
- = 6: External oscillator input divide by 8
- = 7: External oscillator input divide by 16

Option 4: $\overline{\text{RESET}}$ input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 5: L7 Driver
- = 0: Standard TRI-STATE push-pull output
- = 1: Low-current TRI-STATE push-pull output
- = 2: Open-drain TRI-STATE output

Option 6: L6 Driver — (same as option 5)

Option 7: L5 Driver — (same as option 5)

Option 8: L4 Driver — (same as option 5)

Option 9: IN1 input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN2 input — (same as option 9)

Option 11 = 0: V_{CC} Pin — no option available

Option 12: L3 Driver — (same as option 5)

Option 13: L2 Driver — (same as option 5)

Option 14: L1 Driver — (same as option 5)

Option 15: L0 Driver — (same as option 5)

Option 16: SI input — (same as option 9)

Option 17: SO Driver
- = 0: Standard push-pull output
- = 1: Low-current push-pull output
- = 2: Open-drain output

Option List (Continued)

Option 18: SK Driver — (same as option 17)
Option 19: IN0 Input — (same as option 9)
Option 20: IN3 Input — (same as option 9)
Option 21: G0 I/O Port — (same as option 17)
Option 22: G1 I/O Port — (same as option 17)
Option 23: G2 I/O Port — (same as option 17)
Option 24: G3 I/O Port — (same as option 17)
Option 25: D3 Output — (same as option 17)
Option 26: D2 Output — (same as option 17)
Option 27: D1 Output — (same as option 17)
Option 28: D0 Output — (same as option 17)
Option 29: Internal Initialization Logic
= 0: Normal operation
= 1: No internal initialization logic
Option 30: Dual Clock
= 0: Normal operation
= 1: Dual Clock. D0 RC oscillator
= 2: Dual Clock. D0 ext. clock input
} (opt. #28 must = 2)
Option 31: Timer
= 0: Time-base counter
= 1: External event counter
Option 32: Microbus
= 0: Normal
= 1: Microbus (opt. #31 must = 0)
Option 33: COP bonding
(1k and 2K Microcontroller)
= 0: 28-pin package
= 1: 24-pin package
= 2: Same die purchased in both 24 and 28 pin version.
(1K Microcontroller only)
= 3: 20-pin package
= 4: 28- and 20-pin package
= 5: 24- and 20-pin package
= 6: 28-, 24- and 20-pin package

Note:—if opt. #33 = 1 or 2 then opt. #9, 10, 19, 20 and 32 must = 0—if opt. #33 = 3, 4, 5 or 6 then opt. #9, 10, 19, 20, 21, 22, 30 and 32 must = 0.

Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA			OPTION DATA		
OPTION 1 VALUE =	0	IS: GROUND PIN	OPTION 17 VALUE =		IS: SO DRIVER
OPTION 2 VALUE =		IS: CKO PIN	OPTION 18 VALUE =		IS: SK DRIVER
OPTION 3 VALUE =		IS: CKI INPUT	OPTION 19 VALUE =		IS: IN0 INPUT
OPTION 4 VALUE =		IS: RESET INPUT	OPTION 20 VALUE =		IS: IN3 INPUT
OPTION 5 VALUE =		IS: L(7) DRIVER	OPTION 21 VALUE =		IS: G0 I/O PORT
OPTION 6 VALUE =		IS: L(6) DRIVER	OPTION 22 VALUE =		IS: G1 I/O PORT
OPTION 7 VALUE =		IS: L(5) DRIVER	OPTION 23 VALUE =		IS: G2 I/O PORT
OPTION 8 VALUE =		IS: L(4) DRIVER	OPTION 24 VALUE =		IS: G3 I/O PORT
OPTION 9 VALUE =		IS: IN1 INPUT	OPTION 25 VALUE =		IS: D3 OUTPUT
OPTION 10 VALUE =		IS: IN2 INPUT	OPTION 26 VALUE =		IS: D2 OUTPUT
OPTION 11 VALUE =		IS: VCC PIN	OPTION 27 VALUE =		IS: D1 OUTPUT
OPTION 12 VALUE =		IS: L(3) DRIVER	OPTION 28 VALUE =		IS: D0 OUTPUT
OPTION 13 VALUE =		IS: L(2) DRIVER	OPTION 29 VALUE =		IS: INT INIT LOGIC
OPTION 14 VALUE =		IS: L(1) DRIVER	OPTION 30 VALUE =		IS: DUAL CLOCK
OPTION 15 VALUE =		IS: L(0) DRIVER	OPTION 31 VALUE =		IS: TIMER
OPTION 16 VALUE =		IS: SI INPUT	OPTION 32 VALUE =		IS: MICROBUS
			OPTION 33 VALUE =		IS: COP BONDING

National Semiconductor

COP440/COP441/COP442 and COP340/COP341/COP342 Single-Chip N-Channel Microcontrollers

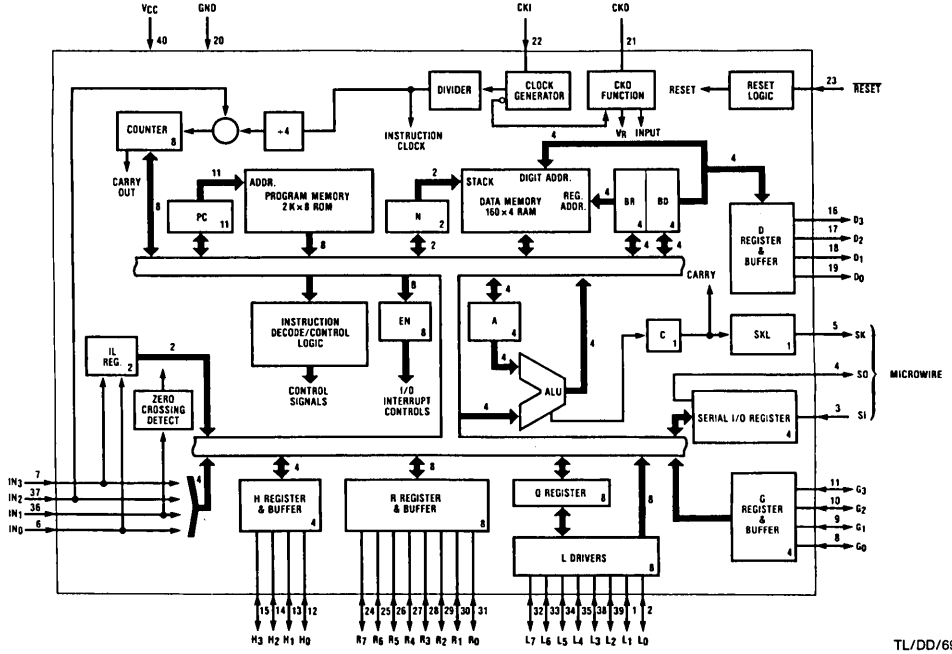
General Description

The COP440, COP441, COP442, COP340, COP341, and COP342 Single-Chip N-Channel Microcontrollers are members of the COPS™ microcontrollers family, fabricated using N-channel, silicon gate MOS technology. These are complete microcontrollers with all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, various output configuration options, and an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and data manipulation. The COP440 is a 40-pin chip and the COP441 is a 28-pin version of the same circuit (12 I/O lines removed). The COP442 is a 24-pin version (4 more input lines removed). The COP340, COP341, COP342 are functional equivalents of the above devices respectively, but operate with an extended temperature range (−40°C to +85°C). Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

Features

- Enhanced, more powerful instruction set
- 2k x 8 ROM, 160 x 4 RAM
- 35 I/O lines (COP440)
- Zero-crossing detect circuitry with hysteresis
- True multi-vectored interrupt from 4 selectable sources (plus restart)
- Four-level subroutine stack (in RAM)
- 4 μs cycle time
- Single supply operation (4.5V–6.3V)
- Programmable time-base counter
- Internal binary counter/register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- LED drive capability
- MICROBUS™ compatible
- Software/hardware compatible with other members of the COP400 family
- Extended temperature range devices COP340, COP341, COP342 (−40°C to +85°C)
- Compatible dual CPU device available (COP2440 series)

Block Diagram



TL/DD/6926–1

FIGURE 1

COP440/COP441/COP442 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Zero-Crossing Detect Pin Relative to GND	−1.2V to +15V
Voltage at Any Other Pin Relative to GND	−0.5V to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	150 mA
Total Sink Current	75 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 3)	4.5	6.3	V
Power Supply Ripple	(Peak to Peak)		0.4	V
Operating Supply Current	(All Inputs and Outputs Open)			
	T_A = 0°C		44	mA
	T_A = 25°C		35	mA
	T_A = 70°C		27	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷16, ÷8)				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
Zero-Crossing Detect Input	See *Figure 7*			
Trip Point		−0.15	0.15	V
Logic High (V_{IH}) Limit			12	V
Logic Low (V_{IL}) Limit		−0.8		V
SO Input Level (Test Mode)	(Note 5)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Capacitance			7.0	pF
Hi-Z Input Leakage		−1.0	+1.0	μA

Note 1: Duty Cycle = $t_{WI}/(t_{WI} + t_{WO})$.

Note 2: See Figure for additional I/O Characteristics.

Note 3: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 4: Exercise great care not to exceed maximum device power dissipation limits when direct-driving LEDs (or sourcing similar loads) at high temperature.

Note 5: SO output "0" level must be less than 0.8V for normal operation.

COP440/COP441/COP442
DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Voltage Levels				
Standard Output				
TTL Operation				
Logic High (V_{OH})	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low (V_{OL})	$I_{OL} = 1.6$ mA		0.4	V
CMOS Operation (Note 1)				
Logic High (V_{OH})	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.4$		V
Logic Low (V_{OL})	$I_{OL} = 10\ \mu A$		0.2	V
Output Current Levels				
Standard Output Source Current	$V_{CC} = 4.5V$, $V_{OH} = 2.4V$	−100	−650	μA
LED Direct Drive Output	$V_{CC} = 6V$, $V_{OH} = 2V$			
Logic High (I_{OH})		−2.5	−17	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
CKO Output				
Oscillator Output Option				
Logic High	$V_{OH} = 2V$	−0.2		mA
Logic Low	$V_{OL} = 0.4V$	0.4		mA
V_R RAM Power Supply Option				
Supply Current	$V_R = 3.3V$		3.0	mA
CKI Sink Current (RC Option)	$V_{IH} = 3.5V$, $V_{CC} = 4.5V$	2.0		mA
Input Current Levels				
Zero-Crossing Detect Input				
Resistance	$V_{IH} = 1.0V$	0.9	4.6	kΩ
Input Load Source Current	$V_{IH} = 2.0V$, $V_{CC} = 4.5V$	14	230	μA
Total Sink Current Allowed				
All I/O Combined			75	mA
Each L, R Port			20	mA
Each D, G, H Port			10	mA
SO, SK			2.5	mA
Total Source Current Allowed				
All I/O Combined			150	mA
L Port			120	mA
L_7–L_4			70	mA
L_3–L_0			70	mA
Each L Pin			23	mA
All Other Output Pins			1.6	mA

Note 1: TRI-STATE and LED configurations are excluded.

COP340/COP341/COP342
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Zero-Crossing Detect Pin Relative to GND	−1.2V to +15V
Voltage at Any Other Pin Relative to GND	−0.5V to +7V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.25W at 85°C
Total Source Current	150 mA
Total Sink Current	75 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 3)	4.5	5.5	V
Power Supply Ripple	(Peak to Peak)		0.4	V
Operating Supply Current	(All Inputs and Outputs Open)			
	$T_A = -40°C$		54	mA
	$T_A = 25°C$		35	mA
	$T_A = 85°C$		25	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷16, ÷8)	V_{CC} = Max	3.0		V
Logic High (V_{IH})		2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
Zero-Crossing Detect Input	See *Figure 7*			
Trip Point		−0.15	0.15	V
Logic High (V_{IH}) Limit			12	V
Logic Low (V_{IL}) Limit		−0.8		V
SO Input Level (Test Mode)	(Note 5)	2.2	2.4	V
All Other Inputs	V_{CC} = Max	3.0		V
Logic High		2.2		V
Logic Low		−0.3	0.6	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Capacitance			7.0	pF
Hi-Z Input Leakage		−2.0	+2.0	μA

Note 1: Duty Cycle = $t_{WI}/(t_{WI} + t_{WO})$.

Note 2: See Figure for additional I/O Characteristics.

Note 3: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 4: Exercise great care not to exceed maximum device power dissipation limits when direct-driving LEDs (or sourcing similar loads) at high temperature.

Note 5: SO output "0" level must be less than 0.6V for normal operation.

COP340/COP341/COP342
DC Electrical Characteristics

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Voltage Levels				
Standard Output				
TTL Operation				
Logic High (V_{OH})	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low (V_{OL})	$I_{OL} = 1.6$ mA		0.4	V
CMOS Operation (Note 1)				
Logic High (V_{OH})	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.5$		V
Logic Low (V_{OL})	$I_{OL} = 10\ \mu A$		0.2	V
Output Current Levels				
Standard Output Source Current	$V_{CC} = 4.5V$, $V_{OH} = 2.4V$	−100	−800	μA
LED Direct Drive Output	$V_{CC} = 5V$ (Note 4)			
Logic High (I_{OH})	$V_{OH} = 2V$	−1.5	−15	mA
TRI-STATE Output Leakage Current		−5.0	+5.0	μA
CKO Output				
Oscillator Output Option				
Logic High	$V_{OH} = 2V$	−0.2		mA
Logic Low	$V_{OL} = 0.4V$	0.4		mA
V_R RAM Power Supply Option				
Supply Current	$V_R = 3.3V$		4.0	mA
CKI Sink Current (RC Option)	$V_{CC} = 4.5V$, $V_{IH} = 3.5V$	2.0		mA
Input Current Levels				
Zero-Crossing Detect Input				
Resistance	$V_{IH} = 1.0V$	0.9	4.6	kΩ
Input Load Source Current	$V_{IH} = 2.0V$, $V_{CC} = 4.5V$	14	280	μA
Total Sink Current Allowed				
All I/O Combined			75	mA
Each L, R Port			20	mA
Each D, G, H Port			10	mA
SO, SK			2.5	mA
Total Source Current Allowed				
All I/O Combined			150	mA
L Port			120	mA
L_7–L_4			70	mA
L_3–L_0			70	mA
Each L Pin			23	mA
All Other Output Pins			1.6	mA

Note 1: TRI-STATE and LED configurations are excluded.

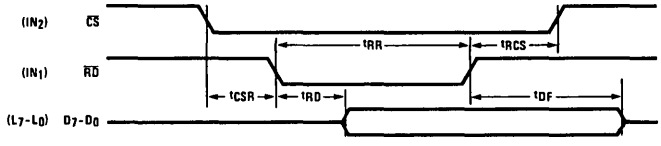
AC Electrical Characteristics

COP440/COP441/COP442: $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

COP340/COP341/COP342: $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

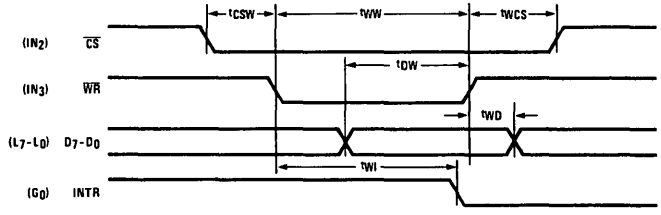
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time–t_E		4.0	10	μs
CKI Frequency	÷16 Mode	1.6	4.0	MHz
	÷8 Mode	0.8	2.0	MHz
	÷4 Mode	0.4	1.0	MHz
Duty Cycle (Note 1)	f_I = 4 MHz	30	60	%
Rise Time	f_I = 4 MHz External Clock		60	ns
Fall Time	f_I = 4 MHz External Clock		40	ns
CKI Using RC *(Figure 9c)*	÷4 Mode			
Frequency	R = 15 kΩ ±5%, C = 100 pF ±10%	0.5	1.0	MHz
Instruction Execution Time–t_E (Note 1)		4.0	8.0	μs
INPUTS: *(Figure 4)*				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		300		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, V_{OUT} = 1.5V			
CKO				
t_{pd1}, t_{pd0}	Crystal Input		0.17	μs
t_{pd1}, t_{pd0}	Schmitt Trigger Input		0.3	μs
SO, SK				
t_{pd1}, t_{pd0}	R_L = 2.4 kΩ		1.0	μs
All Other Outputs	R_L = 5.0 kΩ		1.4	μs
MICROBUS TIMING	C_L = 100 pF, V_{CC} = 5V ±5%			
Read Operation *(Figure 2a)*	TRI-STATE Outputs			
Chip Select Stable Before $\overline{RD}$–t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$–t_{RCS}		20		ns
$\overline{RD}$ Pulse Width–t_{RR}		400		ns
Data Delay from $\overline{RD}$–t_{RD}			375	ns
$\overline{RD}$ to Data Floating–t_{DF}			250	ns
Write Operation *(Figure 2b)*				
Chip Select Stable Before $\overline{WR}$–t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$–t_{WCS}		20		ns
$\overline{WR}$ Pulse Width–t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$–t_{DW}		320		ns
Data Hold Time for $\overline{WR}$–t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$–t_{WI}			700	ns

Note 1: Variation due to the device included.



TL/DD/6926-2

FIGURE 2a. MICROBUS Read Operation Timing

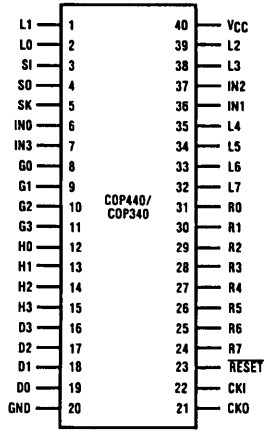


TL/DD/6926-3

FIGURE 2b. MICROBUS Write Operation Timing

Connection Diagrams

Dual-In-Line Package

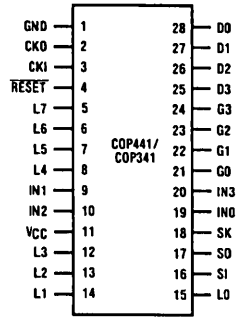


TL/DD/6926–4

Top View

Order Number COP440-XXX/D or COP340-XXX/D
See NS Hermetic Package Number D40C
Order Number COP440-XXX/N or COP340-XXX/N
See NS Molded Package Number N40A

Dual-In-Line Package

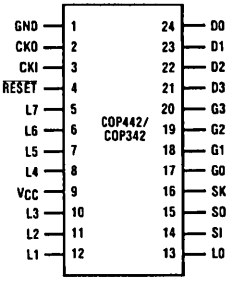


TL/DD/6926–5

Top View

Order Number COP441-XXX/D or COP341-XXX/D
See NS Hermetic Package Number D28C
Order Number COP441-XXX/N or COP341-XXX/N
See NS Molded Package Number N28B

Dual-In-Line Package



TL/DD/6926–6

Top View

Order Number COP442-XXX/D or COP342-XXX/D
See NS Hermetic Package Number D24C
Order Number COP442-XXX/N or COP342-XXX/N
See NS Molded Package Number N24A

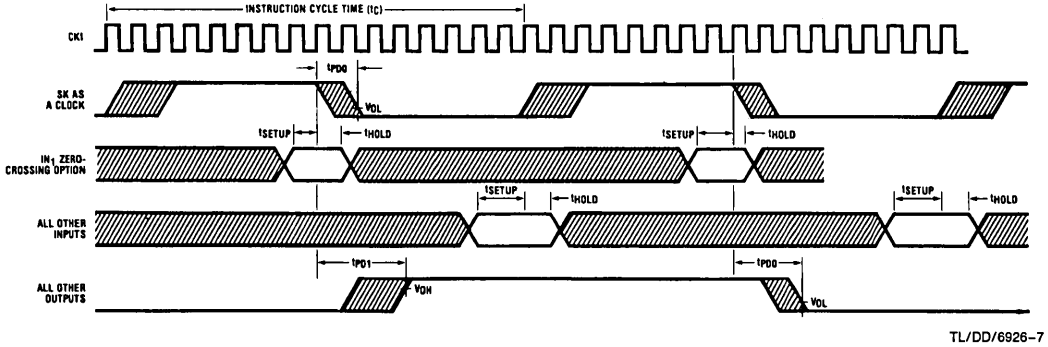
FIGURE 3

Pin Descriptions

Pin	Description
L_7–L_0	8-bit Bidirectional I/O Port with TRI-STATE
G_3–G_0	4-bit Bidirectional I/O Port
D_3–D_0	4-bit General Purpose Output Port
IN_3–IN_0	4-bit General Purpose Input Port (Not Available on COP442/COP342)
SI	Serial Input
SO	Serial Output (or General Purpose Output)
SK	Logic-Controlled Clock (or General Purpose Output)

Pin	Description
CKI	System Oscillator Input
CKO	System Oscillator Output (or General Purpose Input or RAM Power Supply)
$\overline{RESET}$	System Reset Input
V_{CC}	Power Supply
GND	Ground
H_3–H_0	4-bit Bidirectional I/O Port (COP440/COP340 Only)
R_7–R_0	8-Bit Bidirectional I/O Port with TRI-STATE (COP440/COP340 Only)

Timing Diagram



TL/DD/6926–7

FIGURE 4. Input/Output Timing Diagrams (Divide by 16 Mode)

Functional Description

The block diagram of the COP440 is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2.0V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 2,048 byte ROM. As can be seen by an examination of the COP440 instruction set, these words may be program instructions, constants, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, LQID, and LID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by an 11-bit PC register. Its binary value selects one of the 2,048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 640-bit RAM, organized as 10 data registers of 16 4-bit digits. RAM addressing is implemented by an 8-bit B register whose upper 4 bits (Br) select 1 of 10 (0–9) data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into, or from, or exchanged with the A register (accumulator), it may also be loaded into or from the Q latches, L port, R port, EN register, and T counter (internal time base counter). RAM may also be loaded from 4 bits of a ROM word. RAM addressing may also be performed directly to the lower 8 registers by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs. RAM register 8 (Br = 8) also serves as a subroutine stack. Note that it is possible, but not recommended, to alter the contents of the stack by normal data memory access commands.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, N register, to load and input 4 bits of the 8-bit Q latch, EN register, or T counter, to input 4 bits of a ROM word, L or R I/O port data, to input 4-bit G, H, or IN ports, and to perform data exchanges with the SIO register. The accumulator is cleared upon reset.

A 4-bit adder performs the arithmetic and logic functions of the COP440, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

The 8-bit T counter is a binary up counter which can be loaded to and from M and A. The input to this counter is software selectable from two sources: the first coming from a divide-by-four prescaler (from instruction cycle frequency) thus providing a 10-bit time base counter; the second coming from IN_2 input, changing the T counter into an 8-bit external event counter (see EN register below). In this mode, a low-going pulse ("1" to "0") of at least 2 instruction cycles wide will increment the counter. When the counter overflows, an overflow flag will be set (see SKT instruction below) and an interrupt signal will be sent to processor X. The T counter is cleared on reset.

Four general-purpose inputs, IN_3–IN_0, are provided; IN_1, IN_2 and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in MICROBUS applications; IN_1, by another mask-programmable option, can be selected as a true zero-crossing detector with the output triggering an interrupt or being interrogated by an instruction. These two mask-programmable options are mutually exclusive.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G_0 may be mask-programmed as an output for MICROBUS applications.

The H register contents are outputs to a 4-bit general-purpose bidirectional I/O port.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU. Note that unlike most other COPS controllers, Q is cleared on reset.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register. The L I/O port can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The R register, when enabled, outputs to an 8-bit general-purpose, bidirectional, I/O port.

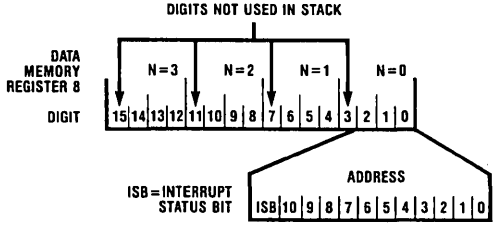
The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register; see EN register description, below). Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream.

The XAS instruction copies the C flag into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the instruction cycle clock.

The 2-bit N register is a stack pointer to the data memory register 8 where the subroutine return address is located. It points to the next location where the address may be stored

Functional Description (Continued)

and increments by 1 after each push of the stack, and decrements by 1 before each pop. The N register can be accessed by exchanging its value with A and is cleared on reset. The stack is 4 addresses deep, 12 bits wide, and does not check for overflow or empty conditions. The RAM digit locations where the addresses are stored are shown in *Figure 5*. The LSBs of the addresses are at digits 0, 4, 8, and 12. The MSBs of digits 2, 6, 10, and 14 contain an interrupt status bit (see Interrupt description, below). The four unused digits (3, 7, 11, and 15) can be used as general data storage. When a subroutine call or interrupt occurs, an 11-bit return address and an interrupt status bit are stored in the stack. The N register is then incremented. When a RET or RETSK instruction is executed, the N register is decremented and then the return address is fetched and loaded into the program counter. The address and interrupt status bits remain in the stack, but will be overwritten when the next subroutine call or interrupt occurs.



TL/DD/6926-8

FIGURE 5. Subroutine Return Address Stack Organization

The EN register in an internal 8-bit register loaded under program control by the LEI instruction (lower 4 bits) or by the CAME instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register.

0. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK output becomes a logic-controlled clock.
1. With EN_1 set, interrupt is enabled with EN_4 and EN_5 selecting the interrupt source. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
2. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN_2 disables the L drivers, placing the L I/O port in a high-impedance input state. A special feature of the COP440 and COP441 is that the MICROBUS option will change the function of this bit to disable any writing into G_0 when EN_2 is set.
3. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0". Table I below provides a summary of the modes associated with EN_3 and EN_0.
4. EN_5 and EN_4 select the source of the interrupt signal.
5. The possible sources are as follows:

EN_5	EN_4	Interrupt Source
0	0	IN_1 (low-going pulse)
0	1	CKO input (if mask-programmed as an input)
1	0	Zero-crossing (or IN_1 level transition)
1	1	T counter overflows

   EN_4 determines the interrupt routine location.
6. With EN_6 set, the internal 8-bit T counter will use IN_2 as its input. With EN_6 reset, the input to the T counter is the output of a divide by four prescaler (from instruction cycle frequency), thus providing a 10-bit time-base counter.
7. With EN_7 set, the R outputs are enabled; if $EN_7 = 0$, the R outputs are disabled.

INTERRUPT

The following features are associated with the interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) together with an interrupt status bit, onto the program counter stack residing in data memory. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset. If EN_4 is reset, the next program address is hex 100; if EN_4 is set, the next program address is hex 300; thus providing a different interrupt location for different interrupt sources.

TABLE I. Enable Register Modes — Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

b. An interrupt will be acknowledged only after the following conditions are met:

1. EN_1 has been set.
2. For an external interrupt input, the signal pulse must be at least two instruction cycles wide.
3. A currently executing instruction has been completed.
4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

c. The instruction at hex address 0FF must be a NOP.

d. A CAME or LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

e. If the interrupt signal source is being changed, the interrupt must be disabled prior to, or at, the same time with the change to avoid false interrupts. An interrupt may be enabled only if the interrupt source is not changing. A sample code for changing the interrupt source and enabling the interrupt is as follows:

```
CAME        ; disable interrupt & alter interrupt source
SMB  1      ; set interrupt enable bit
CAME        ; enable interrupt
```

f. An interrupt status bit is stored together with the return address in the stack. The status bit is set if an interrupt occurs at a point in the program where the next instruction is to be skipped; upon returning from the interrupt routine, this set status bit will cause the next instruction to be skipped. Subroutine and interrupt nesting inside interrupt routines are allowed. Note that this differs from the COP420/420C/420L/444L series.

MICROBUS INTERFACE
(not available in COP442, COP342)

The COP440 series has an option which allows them to be used as peripheral microprocessor devices, inputting and outputting data from and to a host microprocessor (μP). IN_1, IN_2 and IN_3 general purpose inputs become MICROBUS-compatible read-strobe, chip-select, and write-strobe lines, respectively. IN_1 becomes $\overline{RD}$—a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN_2 becomes $\overline{CS}$—a logic "0" on this line selects the COPS processor as the μP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN_3 becomes $\overline{WR}$—a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COPS processor. G_0 becomes INTR, a "ready" output, reset by a write pulse from the μP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COPS processor. G_0 output can be separated from other G outputs by the EN_2 bit (see EN description above).

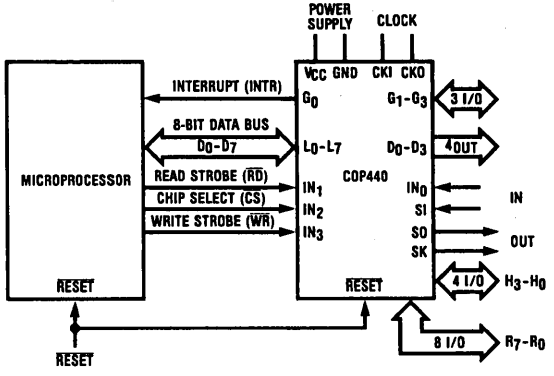
This option has been designed for compatibility with National's MICROBUS—a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS National Publication.) The functional and timing relationships between the COPS processor signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams *(Figure 2)*. Connection of the COP440 to the MICROBUS is shown in *Figure 6*.

Note: TRI-STATE outputs must be used on L port.

ZERO-CROSSING DETECTION
(not available on the COP442, COP342)

The following features are associated with the IN_1 pin: ININ and INIL instructions input the state of IN_1 to A_1; IN_1 interrupt generates an interrupt pulse when a low-going transition ("1" to "0") occurs on IN_1; zero-crossing interrupt generates an interrupt pulse when an IN_1 transition occurs (both "1" to "0" and "0" to "1").

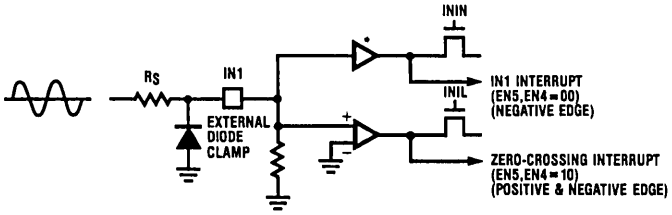
If the zero-crossing detector is mask-programmed in (see *Figure 7a*), the INIL instruction and zero-crossing interrupt will input the state of IN_1 through the true zero-crossing detector ("1" if input > 0V, "0" if input < 0V). The ININ instruction and IN_1 interrupt will then have unique logic HIGH and LOW levels depending on the IN port input level chosen. If normal (TTL) level is chosen, logic HIGH level is 3.0V (3.3V for COP340/341) and logic LOW level is 0.8V



TL/DD/6926–9

FIGURE 6. MICROBUS Option Interconnect

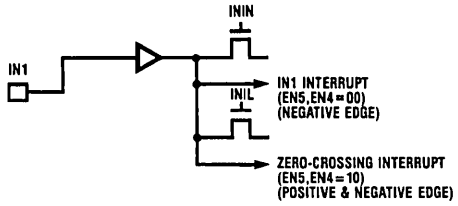
Functional Description (Continued)



TL/DD/6926-10

*Note: This input has a different set of logic HIGH and LOW levels; see above description.

a. Zero-Crossing Detect Logic Option



TL/DD/6926-11

b. IN, without Zero-Crossing Detect Logic

FIGURE 7. IN, Mask-Programmable Options

(0.6V for COP340/341); if high trip level is chosen, logic HIGH level is 5.4V and logic LOW level is 1.2V. If the zero-crossing detector is not mask-programmed in (see *Figure 7b*), IN_1 will have logic HIGH and LOW levels that are defined for the IN port (see option list).

The zero-crossing detector input contains a small hysteresis (50 mV typical) to eliminate signal noise, and is not a high impedance input but contains a resistive load to ground. Since this input can withstand a voltage range of −0.8V to +12V, an external clamping diode is needed for most input signals, as shown in *Figure 7a*, to limit the voltage below ground. An external resistor, R_S may be needed for the following two cases:

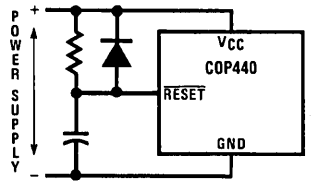
a. Input signal exceeds 12V; R_S and the internal resistor act as a voltage divider to reduce the voltage at the input pin to below 12V.

b. Signal comes from a low impedance source; when the voltage at the pin is clamped to −0.7V by the forward bias voltage of an external diode, R_S limits the current going through the diode.

INITIALIZATION

The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times. The user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as in *Figure 8*. The external POR (Power-on-Reset) delay must be greater than the internal POR. The internal POR delay is 2600 internal clock cycles.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, G, H, IL, L, N, Q, R, and T registers are cleared. The SK output is enabled as a SYNC output by setting the SKL latch, thus providing a clock. RAM (data memory and stack) is not cleared. The first instruction at address 0 must be a CLRA.



TL/DD/6926-12

RC ≥ 5 × power supply rise time

FIGURE 8. Power-Up Clear Circuit

OSCILLATOR

There are three basic clock oscillator configurations available, as shown by *Figure 9*.

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The cycle frequency equals the crystal frequency divided by 16 (optional by 8). Thus a 4 MHz crystal with the divide-by-16 option selected will give a 250 kHz cycle frequency (4 μs instruction cycle time).

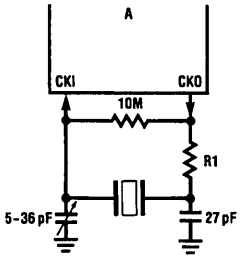
b. External Oscillator. CKI is an external clock input signal. The external frequency is divided by 16 (optional by 8 or 4) to give the cycle frequency. If the divide-by-4 option is selected, the CKI input level is the Schmitt-trigger level. CKO is now available to be used as the RAM power supply (V_R) or as a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The cycle frequency equals the oscillation frequency divided by 4. CKO is available for non-timing functions.

CKO PIN OPTIONS

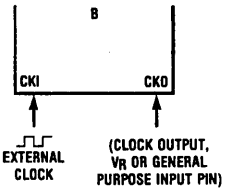
As an option, CKO can be an oscillator output. In a crystal controlled oscillator system, this signal is used as an output to the crystal network. As another option, CKO can be an interrupt input or a general purpose input, reading into bit 2 of A (accumulator) through the INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing

Functional Description (Continued)



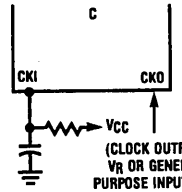
TL/DD/6926-13

a. Crystal Oscillator



TL/DD/6926-14

b. External Oscillator



TL/DD/6926-15

c. RC Controlled Oscillator

Crystal Oscillator

Crystal Value	R_1
4 MHz	1k
3.58 MHz	1k
2.10 MHz	2k

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Execution Time (μs)
13	100	5.0 ±20%
6.8	220	5.3 ±23%
8.2	300	8.0 ±22%
22	100	8.2 ±17%

Note: 5 kΩ ≤ R ≤ 50 kΩ
50 pF ≤ C ≤ 360 pF

FIGURE 9. COP440/441/442 Oscillators

its connection to a standby/backup power supply to maintain the data integrity of RAM registers 0–3 with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either of the two latter options is appropriate in applications where the system configuration does not require use of the CKO pin for timing functions.

RAM KEEP-ALIVE OPTION

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the lower 4 registers of the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. When V_{CC} is on, V_R must be within the operating voltage range of the chip, and within 1V of V_{CC}.
3. V_R must be ≥ 3.3V with V_{CC} off.

I/O OPTIONS

COP440 inputs have the following optional configurations, illustrated in *Figure 10*.

a. An on-chip depletion load device to V_{CC}.

b. A Hi-Z input which must be driven to a "1" or "0" by external components.

c. A resistive load to GND for the zero-crossing input option (IN_1 only).

COP440 outputs have the following optional configurations:

d. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements. Available on SO, SK, D, G, and H outputs.

e. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, D, G, L, H, and R outputs.

f. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

g. Standard L,R—same as **d.**, but may be disabled. Available on L and R outputs only (disabled on reset).

h. LED Direct Drive—an enhancement-mode device to ground and V_{CC} together with a depletion device to V_{CC} meeting the typical current sourcing requirements of the segments of an LED display. The sourcing devices are clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the output in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note 1: When the driver is disabled, the depletion device may cause the output to settle down to an intermediate level between V_{CC} and GND. This voltage cannot be relied upon as a "1" level when reading the L inputs. The external signal must drive it to a "1" level.

Note 2: Much power is dissipated by this driver in driving an LED. Care must be taken to limit the power dissipation of the chip to within the absolute maximum ratings specified.

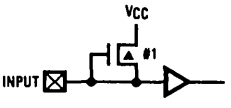
I. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L and R outputs only (in TRI-STATE mode on reset).

j. Push-Pull R—same as **f.**, but may be disabled. Available on R outputs only.

k. Additional depletion pull-up—a depletion load to V_{CC} with the same current sourcing capability as the input load **a.**, in addition to the output drive chosen. Available on L and R outputs only. *This device cannot be disabled*; therefore, open-drain outputs with "1" output and TRI-STATE outputs do not show high-impedance characteristics. This device is useful in applications where a pull-up with low source current is desired, e.g., reading keyboards and switches.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6 respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figures 11* and *12* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP440 system.

Functional Description (Continued)

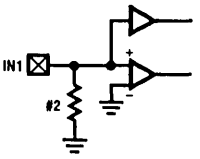


TL/DD/6926-16

a. Input with Load

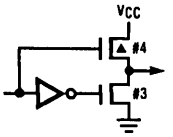
TL/DD/6926-17

b. Hi-Z Input



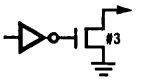
TL/DD/6926-18

c. Zero-Crossing Input



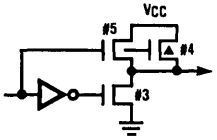
TL/DD/6926-19

d. Standard Output



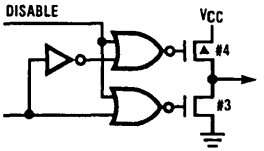
TL/DD/6926-20

e. Open-Drain Output



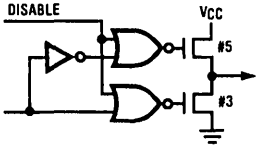
TL/DD/6926-21

f. Push-Pull Output



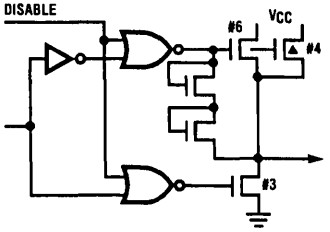
TL/DD/6926-22

g. Standard L, R Outputs



TL/DD/6926-23

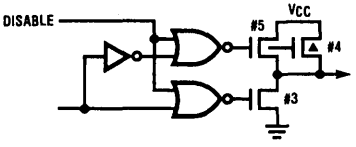
I. TRI-STATE Push-Pull (L, R) Outputs



TL/DD/6926-24

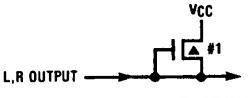
(▲ is depletion device)

h. LED (L) Outputs



TL/DD/6926-25

j. Push-Pull R Outputs



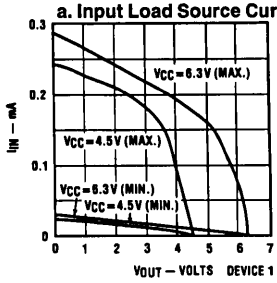
TL/DD/6926-26

k. Additional L, R Outputs Pull-Up

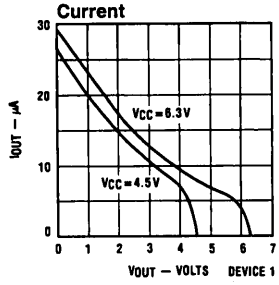
FIGURE 10. Input/Output Configurations

Typical Performance Characteristics

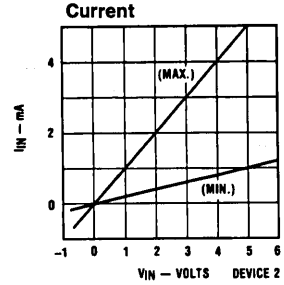
a. Input Load Source Current



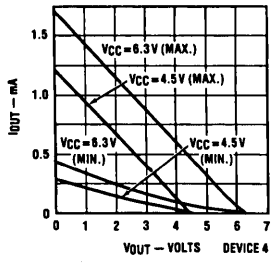
b. Input Load Minimum Source Current



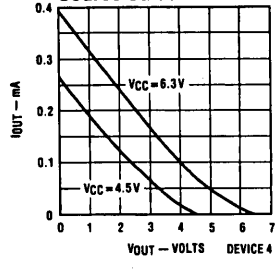
c. Zero-Crossing Detect Input Current



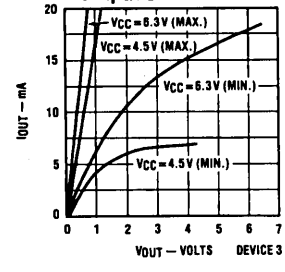
d. Standard Output Source Current



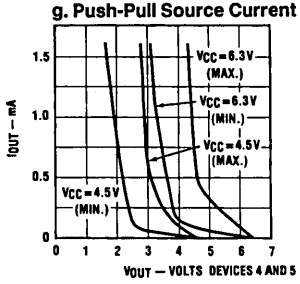
e. Standard Output Minimum Source Current



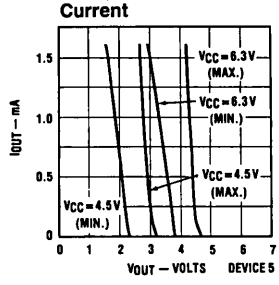
f. Output Sink Current



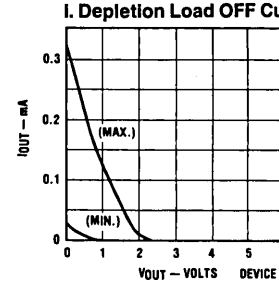
g. Push-Pull Source Current



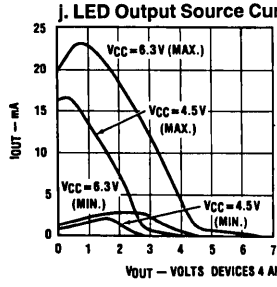
h. TRI-STATE Output Source Current



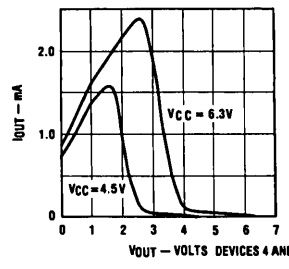
I. Depletion Load OFF Current



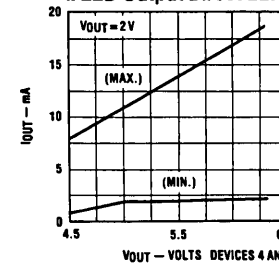
j. LED Output Source Current



k. LED Output Minimum Source Current



I. LED Output Direct LED Drive

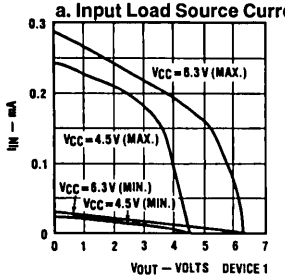


TL/DD/6926–27

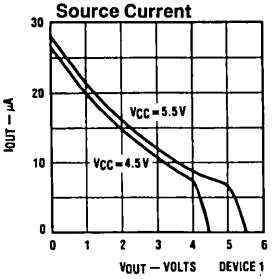
FIGURE 11. COP440/441/442 I/O Characteristics

Typical Performance Characteristics (Continued)

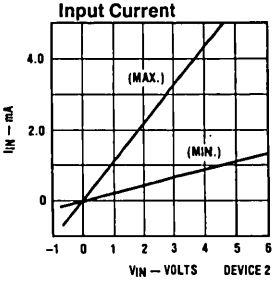
a. Input Load Source Current



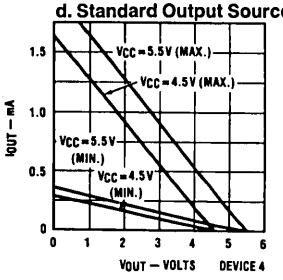
b. Input Load Minimum Source Current



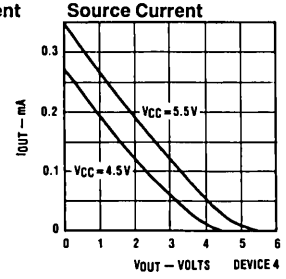
c. Zero-Crossing Detect Input Current



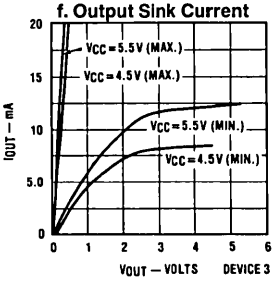
d. Standard Output Source Current



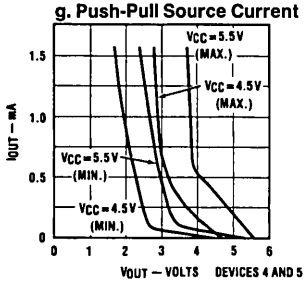
e. Standard Output Minimum Source Current



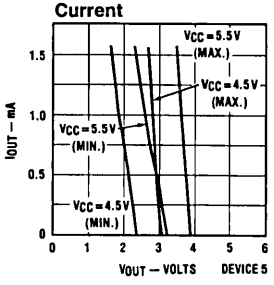
f. Output Sink Current



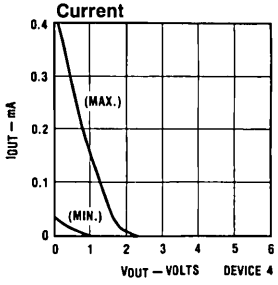
g. Push-Pull Source Current



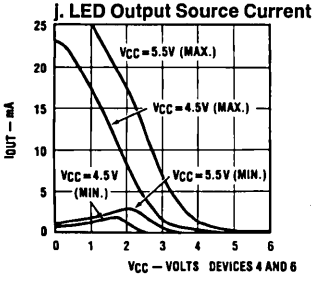
h. TRI-STATE Output Source Current



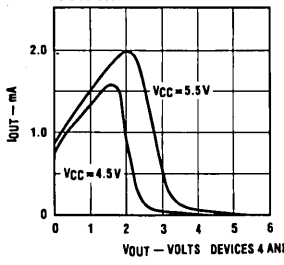
I. Depletion Load OFF Current



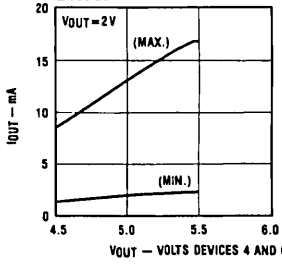
j. LED Output Source Current



k. LED Output Minimum Source Current



l. LED Output Direct LED Driver



TL/DD/6926–28

FIGURE 12. CCOP340/341/342 I/O Characteristics

Power Dissipation

In order not to damage the device by exceeding the absolute maximum power dissipation rating, the amount of power dissipated inside the chip must be carefully controlled. As an example, an application uses a COP440 in room temperature (25°C) environment with a V_{CC} power supply of 6V; IN and SI inputs have internal loads; G and D ports drive loads that may sink up to 2 mA into the chip; H port with standard output option reads switches; L port with the LED option drives a multiplexed seven-segment display; R, SO and SK drive MOS inputs that do not source or sink any current.

a. At 25°C, maximum power dissipation allowed = 750 mW

b. Power dissipation by chip except

$$I/O = I_{CC} \times V_{CC} = 35\ mA \times 6V = 210\ mW$$

c. Maximum power dissipation by IN,

$$SI = 5 \times 0.3\ mA \times 6V = 9\ mW$$

d. G and D ports are sinking current from external loads; maximum output voltage with 2 mA sink current is less than 0.4V. Power dissipation by G and D ports =

$$2\ mA \times 0.4V \times 8 = 6.4\ mW$$

e. Maximum power dissipation by H port =

$$4 \times 1.5\ mA \times 6V = 36\ mW$$

f. When the seven segments of the LED are turned on, the output voltage is about 2V, so that the segment current is 17 mA. Power dissipation by L port =

$$7 \times 17\ mA \times (6V - 2V) = 476\ mW$$

This power dissipation caused by driving LEDs is usually the highest among the various sources.

g. R, SO, and SK do not dissipate any significant amount of power because they do not need to source or sink any current.

Total power dissipation (TPD) inside the device is the sum of items b through g above.

$$TPD = 210 + 9 + 6 + 36 + 476\ mW = 737\ mW$$

This is within the 750 mW limit at room temperature. If this application has to operate at 70°C, then the power dissipation must be reduced to meet the limit at that temperature. Some ways to achieve this would be to limit the LED current or to use an external LED driver.

At 70°C the absolute maximum power dissipation rating drops to 400 mW. The user must be careful not to exceed this value.

COP440 SERIES DEVICES

If the COP440 is bonded as a 28- or 24-pin device, it becomes the COP441 or COP442, respectively, as illustrated in *Figure 3*. Note that the COP441 and COP442 do not include H and R ports. In addition, the COP442 does not include IN inputs; use of this option precludes the use of the IN options, the interrupt feature with IN as input, the zero-crossing detect option, IN_2 external event counter input, and the MICROBUS option. All other options are available.

COP340, COP341, and COP342 are extended temperature versions of the COP440, COP441, and COP442, respectively.

COP440 Series Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP440 series instruction set.

TABLE II. COP440 Series Instruction Set Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	8-bit RAM Address Register
Br	Upper 4 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	8-bit Enable Register
G	4-bit Register to latch data for G I/O Port
H	4-bit Register to latch data for H I/O Port
IL	Two 1-bit Latches associated with the IN_3 or IN_0 Inputs
IN	4-bit Input Port
IN_1Z	Zero-Crossing Input
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
N	2-bit subroutine return address stack pointer
PC	11-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
R	8-bit Register to latch data for R TRI-STATE I/O Port
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output
T	8-bit Binary Counter Register

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	4-bit Operand Field, 0–9 binary (RAM Register Select)
a	11-bit Operand Field, 0–2047 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Content of RAM location addressed by s
RAM_N	Content of RAM location addressed by stack pointer N
ROM(t)	Content of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
⟷	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values
V	OR

Instruction Set

TABLE III. COP440 Series Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC/LOGIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
OR		33 1A	|0011|0011| |0001|1010|	A ∨ M → A	None	OR RAM with A
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM ($PC_{10:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– – –	|0110|0|$a_{10:8}$| |$a_{7:0}$|	a → PC	None	Jump
JP	a	– –	|1| $a_{6:0}$ | (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 4)
		– –	|11| $a_{5:0}$ | (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	|10| $a_{5:0}$ |	PC + 1 → RAM_N N + 1 → N 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	|0110|1|$a_{10:8}$| |$a_{7:0}$|	PC + 1 → RAM_N N + 1 → N a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	N – 1 → N RAM_N → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	N – 1 → N RAM_N → PC	Always Skip on Return	Return from Subroutine then Skip

Instruction Set (Continued)

TABLE III. COP440 Series Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAME		33 1F	0011 | 0011 0001 | 1111	A → $EN_{7:4}$ RAM(B) → $EN_{3:0}$	None	Copy A, RAM to EN
CAMQ		33 3C	0011 | 0011 0011 | 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CAMT		33 3F	0011 | 0011 0011 | 1111	A → $T_{7:4}$ RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CEMA		33 0F	0011 | 0011 0000 | 1111	$EN_{7:4}$ → RAM(B) $EN_{3:0}$ → A	None	Copy EN to RAM, A
CQMA		33 2C	0011 | 0011 0010 | 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
CTMA		33 2F	0011 | 0011 0010 | 1111	$T_{7:4}$ → RAM(B) $T_{3:0}$ → A	None	Copy T to RAM, A
LD	r	-5	00 | r | 0101 r = 0:3	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 --	00 | 10 | 0011 0 | r | d r = 0:7	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LID		33 19	0011 | 0011 0001 | 1001	ROM ($PC_{10:8}$, A, M) → M, A	None	Load RAM, A Indirect
LQID		BF	1011 | 1111	ROM($PC_{10:8}$,A,M) → Q	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 | 1100 0100 | 0101 0100 | 0010 0100 | 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 | 1101 0100 | 0111 0100 | 0110 0100 | 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7-	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 | r | 0110 r = 0:3	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 | 0011 1 | r | d r = 0:7	RAM(r,d) ⟷ A	None	Exchange A with RAM pointed to directly by r,d
XDS	r	-7	00 | r | 0111 r = 0:3	RAM(B) ⟷ A Bd - 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	00 | r | 0100 r = 0:3	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

Instruction Set (Continued)

TABLE III. COP440 Series Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	--	|00|r|(d − 1)| r = 0:3, d = 0,9:15	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
			or			
		33	|0011|0011|			
		--	|1| r | d| r = 0:7, any d			
LEI	y	33	|0011|0011|	y → $EN_{3:0}$	None	Load lower half of EN Immediate
		6-	|0110| y |			
XABR		12	|0001|0010|	A ⟷ Br	None	Exchange A with Br
XAN		33	|0011|0011|	A ⟷ N(0,0 → A_3, A_2)	None	Exchange A with N
		0B	|0000|1011|			
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	|0011|0011|		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
		21	|0010|0001|			
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	G_0 = 0	
	1	11	|0001|0001|		G_1 = 0	
	2	03	|0000|0011|		G_2 = 0	
	3	13	|0001|0011|		G_3 = 0	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1$ = 0	
	2	03	|0000|0011|		$RAM(B)_2$ = 0	
	3	13	|0001|0011|		$RAM(B)_3$ = 0	
SKSZ		33	|0011|0011|		SIO = 0	Skip if SIO is Zero
		1C	|0001|1100|			
SKT		41	|0100|0001|		T counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP440 Series Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
CAMR		33 3D	0011 | 0011 0011 | 1101	$A \rightarrow R_{7:4}$ $RAM(B) \rightarrow R_{3:0}$	None	Output A, RAM to R Port
ING		33 2A	0011 | 0011 0010 | 1010	$G \rightarrow A$	None	Input G Port to A
INH		33 2B	0011 | 0011 0010 | 1011	$H \rightarrow A$	None	Input H Port to A
ININ		33 28	0011 | 0011 0010 | 1000	$IN \rightarrow A$	None	Input IN Inputs to A (Note 2)
INIL		33 29	0011 | 0011 0010 | 1001	IL_3, CKO, IN_1Z, $IL_0 \rightarrow A$	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 | 0011 0010 | 1110	$L_{7:4} \rightarrow RAM(B)$ $L_{3:0} \rightarrow A$	None	Input L Port to RAM, A
INR		33 2D	0011 | 0011 0010 | 1101	$R_{7:4} \rightarrow RAM(B)$ $R_{3:0} \rightarrow A$	None	Input R Port to RAM,A
OBD		33 3E	0011 | 0011 0011 | 1110	$Bd \rightarrow D$	None	Output Bd to D Port
OGI	y	33 5–	0011 | 0011 0101 | y	$y \rightarrow G$	None	Output to G Port Immediate
OMG		33 3A	0011 | 0011 0011 | 1010	$RAM(B) \rightarrow G$	None	Output RAM to G Port
OMH		33 3B	0011 | 0011 0011 | 1011	$RAM(B) \rightarrow H$	None	Output RAM to H Port
XAS		4F	0100 | 1111	$A \longleftrightarrow SIO$, $C \rightarrow SKL$	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin COP442/COP342 since this device does not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (00010 is loaded into the upper 5 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP440 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN register, above). If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

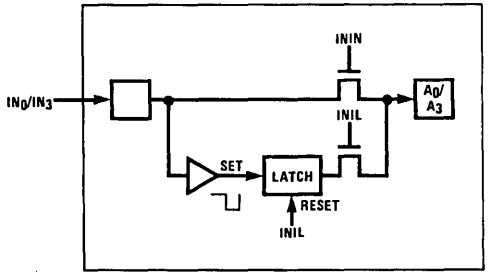
JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, $PC_{10:8}$, A, M. PC_{10}, PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles if executed, 1 instruction cycle time if skipped.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0, CKO and IN_1 into A (see *Figure 13*). The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask-programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. Unlike the COP420/420C/420L/444L series, INIL will input IN_1 into A1.



TL/DD/6926-29

FIGURE 13. INIL Hardware Implementation

If zero-crossing detect is selected, the IN_1 input will go through the detection logic, thus allowing the user to interrogate the input, sending a "1" if the input is above 0V and a "0" if it is below 0V. INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction. It is also useful in checking the status of the zero-crossing detect input. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction, and the IN_1 input does not go through zero-crossing logic so that it has the same logic level as the other IN inputs for the ININ instruction (see *Figure 9*).

Note: IL latches are cleared on reset. This is different from the COP420/420C/420L/444L series.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC_{10}:PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. Note that LQID takes two instruction cycles if executed and one instruction cycle if skipped. Unlike most other COPS processors, this instruction does not push the stack.

LID INSTRUCTION

LID (Load Indirect) loads M and A with the contents of ROM pointed to by the 11-bit word PC_{10}:PC_8, A, M. Note that LID takes three instruction cycles if executed and two if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter (see internal logic, above) overflow latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

INSTRUCTION SET NOTES

a. The first word of a COP440 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from program memory. Thus program paths take the same number of cycle times whether instructions are skipped or executed, except for LID, LQID, and JID.

c. The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, LQID, or LID instruction is the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23, 27, or 31 will access data in the next group of four pages.

Option List

The COP440 mask-programmable options are assigned numbers which correspond with the COP440 pins.

Option 1: L_1 I/O Port (see note below)
- = 0: Standard output
- = 1: Open-drain output
- = 2: LED direct drive output
- = 3: TRI-STATE output
- = 4: same as 0 with extra load device to V_{CC}
- = 5: same as 1 with extra load device to V_{CC}
- = 6: same as 2 with extra load device to V_{CC}
- = 7: same as 3 with extra load device to V_{CC}

Option 2: L_0 I/O Port
(same as Option 1)

Option 3: SI Input
- = 0: Input with load device to V_{CC}
- = 1: Hi-Z Input

Option 4: SO Output
- = 0: Standard output
- = 1: Open-drain output
- = 2: Push-pull output

Option 5: SK Output
(same as Option 4)

Option 6: IN_0 Input
(same as Option 3)

Option 7: IN_3 Input
(same as Option 3)

Option 8: G_0 I/O Port
- = 0: Standard output
- = 1: Open-drain output

Option 9, G_1 I/O Port
(same as Option 8)

Option 10: G_2 I/O Port
(same as Option 8)

Option 11: G_3 I/O Port
(same as Option 8)

Option 12: H_0 I/O Port
(same as Option 8)

Option 13: H_1 I/O Port
(same as Option 8)

Option 14: H_2 I/O Port
(same as Option 8)

Option 15: H_3 I/O Port
(same as Option 8)

Option 16: D_3 Output
(same as Option 8)

Option 17: D_2 Output
(same as Option 8)

Option 18: D_1 Output
(same as Option 8)

Option 19: D_0 Output
(same as Option 8)

Option 20: GND—No options available

Option 21: CKO Pin
- = 0: Oscillator output
- = 1: RAM power supply (V_R) input
- = 2: General purpose input with load device to V_{CC}
- = 3: General purpose Hi-Z input

Option 22: CKI Input
- = 0: Crystal input divided by 16
- = 1: Crystal input divided by 8
- = 2: Single-pin RC controlled oscillator (÷4)
- = 3: Schmitt trigger clock input (÷4)

Option 23: $\overline{\text{RESET}}$ Input
(same as Option 3)

Option 24: R_7 I/O Port (see note below)
- = 0: Standard output
- = 1: Open-drain output
- = 2: Push-pull output
- = 3: TRI-STATE output
- = 4: same as 0 with extra load device to V_{CC}
- = 5: same as 1 with extra load device to V_{CC}
- = 6: same as 2 with extra load device to V_{CC}
- = 7: same as 3 with extra load device to V_{CC}

Option 25: R_6 I/O Port
(same as Option 24)

Option 26: R_5 I/O Port
(same as Option 24)

Option 27: R_4 I/O Port
(same as Option 24)

Option 28: R_3 I/O Port
(same as Option 24)

Option 29: R_2 I/O Port
(same as Option 24)

Option 30: R_1 I/O Port
(same as Option 24)

Option 31: R_0 I/O Port
(same as Option 24)

Option 32: L_7 I/O Port
(same as Option 1)

Option 33: L_6 I/O Port
(same as Option 1)

Option 34: L_5 I/O Port
(same as Option 1)

Option 35: L_4 I/O Port
(same as Option 1)

Option 36: IN_1 Input
- = 0: Input with load device to V_{CC}
- = 1: Hi-Z Input
- = 2: Zero-crossing detect input (Option 41 = 0)

Option 37: IN_2 Input
(same as Option 3)

Option 38: L_3 I/O Port
(same as Option 1)

Option 39: L_2 I/O Port
(same as Option 1)

Option 40: V_{CC}—no options available

Option List (Continued)

Option 41: COP Function
= 0: Normal
= 1: MICROBUS option

Option 42: IN Input Levels
= 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
= 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 43: G Input Levels
(same as Option 42)

Option 44: L Input Levels
(same as Option 42)

Option 45: CKO Input Levels
(same as Option 42)

Option 46: SI Input Levels
(same as Option 42)

Option 47: R Input Levels
(same as Option 42)

Option 48: H Input Levels
(same as Option 42)

Option 49: No option available

Option 50: COP Bonding
= 0: COP440 (40-pin device)
= 1: COP441 (28-pin device)
= 2: COP442 (24-pin device)
= 3: COP440 and COP441
= 4: COP440 and COP442
= 5: COP440, COP441, and COP442
= 6: COP441 and COP442

COP440 Option Table

The following options information is to be sent to National along with the EPROM.

OPTION 1 VALUE = _____ IS: L_1 I/O PORT
OPTION 2 VALUE = _____ IS: L_0 I/O PORT
OPTION 3 VALUE = _____ IS: SI INPUT
OPTION 4 VALUE = _____ IS: SO OUTPUT
OPTION 5 VALUE = _____ IS: SK OUTPUT
OPTION 6 VALUE = _____ IS: IN_0 INPUT
OPTION 7 VALUE = _____ IS: IN_3 INPUT
OPTION 8 VALUE = _____ IS: G_0 I/O PORT
OPTION 9 VALUE = _____ IS: G_1 I/O PORT
OPTION 10 VALUE = _____ IS: G_2 I/O PORT
OPTION 11 VALUE = _____ IS: G_3 I/O PORT
OPTION 12 VALUE = _____ IS: H_0 I/O PORT
OPTION 13 VALUE = _____ IS: H_1 I/O PORT
OPTION 14 VALUE = _____ IS: H_2 I/O PORT
OPTION 15 VALUE = _____ IS: H_3 I/O PORT
OPTION 16 VALUE = _____ IS: D_3 OUTPUT
OPTION 17 VALUE = _____ IS: D_2 OUTPUT
OPTION 18 VALUE = _____ IS: D_1 OUTPUT
OPTION 19 VALUE = _____ IS: D_0 OUTPUT
OPTION 20 VALUE = 0 IS: GROUND PIN
OPTION 21 VALUE = _____ IS: CKO PIN
OPTION 22 VALUE = _____ IS: CKI INPUT
OPTION 23 VALUE = _____ IS: RESET INPUT
OPTION 24 VALUE = _____ IS: R_7 I/O PORT
OPTION 25 VALUE = _____ IS: R_6 I/O PORT
OPTION 26 VALUE = _____ IS: R_5 I/O PORT
OPTION 27 VALUE = _____ IS: R_4 I/O PORT
OPTION 28 VALUE = _____ IS: R_3 I/O PORT
OPTION 29 VALUE = _____ IS: R_2 I/O PORT
OPTION 30 VALUE = _____ IS: R_1 I/O PORT
OPTION 31 VALUE = _____ IS: R_0 I/O PORT
OPTION 32 VALUE = _____ IS: L_7 I/O PORT
OPTION 33 VALUE = _____ IS: L_6 I/O PORT
OPTION 34 VALUE = _____ IS: L_5 I/O PORT
OPTION 35 VALUE = _____ IS: L_4 I/O PORT
OPTION 36 VALUE = _____ IS: IN_1 INPUT
OPTION 37 VALUE = _____ IS: IN_2 INPUT
OPTION 38 VALUE = _____ IS: L_3 I/O PORT
OPTION 39 VALUE = _____ IS: L_2 I/O PORT
OPTION 40 VALUE = 0 IS: V_{CC}
OPTION 41 VALUE = _____ IS: COP FUNCTION
OPTION 42 VALUE = _____ IS: IN INPUT LEVELS
OPTION 43 VALUE = _____ IS: G INPUT LEVELS
OPTION 44 VALUE = _____ IS: L INPUT LEVELS
OPTION 45 VALUE = _____ IS: CKO INPUT LEVELS
OPTION 46 VALUE = _____ IS: SI INPUT LEVELS
OPTION 47 VALUE = _____ IS: R INPUT LEVELS
OPTION 48 VALUE = _____ IS: H INPUT LEVELS
OPTION 49 VALUE = _____ IS: NO OPTION
OPTION 50 VALUE = _____ IS: COP BONDING

Note on L and R I/O Port Options

If L and R I/O Ports are used as inputs, the following must be observed:

a. Open-Drain output (selection 1) is allowed only if external pull-up is provided.

b. If L and R output ports are disabled when reading, an external pull-up is required unless selections 4, 5, 6, or 7 are chosen.

c. If L output port is enabled, selections 3 and 7 are not allowed.

d. If R output port is enabled, selections 2, 3, 6, and 7 are not allowed.

Test Mode (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP440. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

National Semiconductor

COP444L/COP445L/COP344L/COP345L Single-Chip N-Channel Microcontrollers

General Description

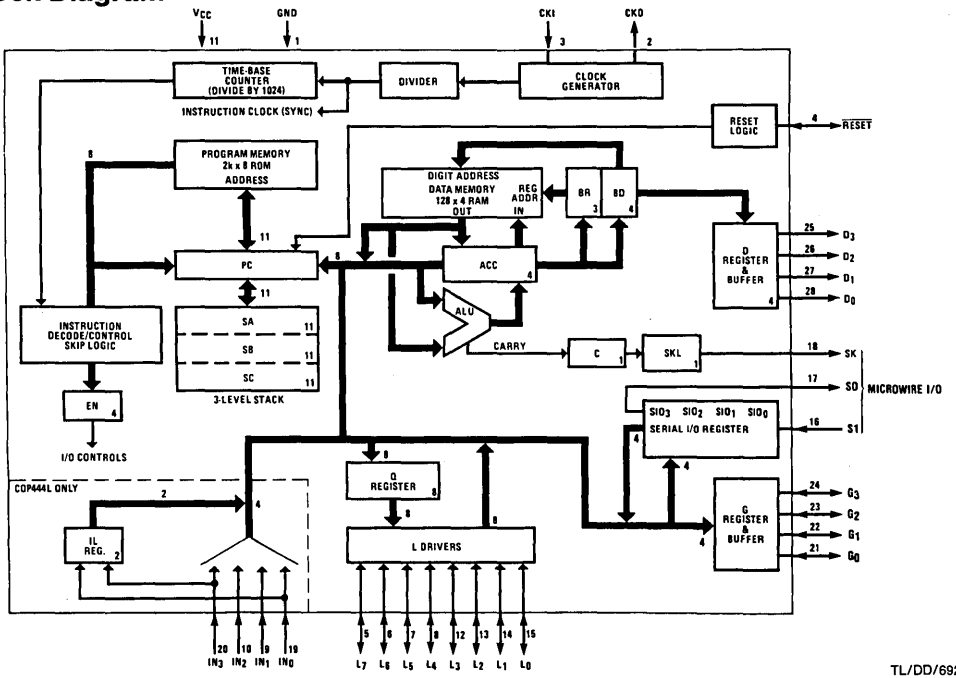
The COP444L, COP445L, COP344L, and COP345L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These controller oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP445L is identical to the COP444L, but with 19 I/O lines instead of 23. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

The COP344L and COP345L are exact functional equivalents, but extended temperature range versions of the COP444L and COP445L respectively.

Features

- Low cost
- Powerful instruction set
- 2k x 8 ROM, 128 x 4 RAM
- 23 I/O lines (COP444L)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 15 μs instruction time
- Single supply operation (4.5–6.3V)
- Low current drain (11 mA max.)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range devices COP344L/COP345L (−40°C to +85°C)
- Wider supply range (4.5–9.5V) optionally available

Block Diagram



TL/DD/6928–1

FIGURE 1

COP444L/COP445L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	0.75 Watt at 25°C
	0.4 Watt at 70°C
Total Source Current	120 mA
Total Sink current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolulte maximum ratings.

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Optional Operating Voltage (V_{CC})		4.5	9.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		13	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷32, ÷16, ÷8)				
Logic High (V_{IH})	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0	0.4	V
Logic Low (V_{IL})		−0.3		
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 3)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±10%	−0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±5%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)				
Logic High	I_{OH} = −10 μA	V_{CC}−1		V
Logic Low	I_{OL} = +10μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.8V for normal operation.

COP444L/COP445L (Continued)

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted. (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	1.8		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	1.2		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.9		mA
L_0–L_7 Outputs and Standard	$V_{CC} = 9.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3, D_0–D_3 Outputs (I_{OL})	$V_{CC} = 6.3V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 9.5V, V_{OL} = 1.0V$	15		mA
High Current Options (I_{OL})	$V_{CC} = 6.3V, V_{OL} = 1.0V$	11		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7.5		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 9.5V, V_{OL} = 1.0V$	30		mA
Very High Current Options (I_{OL})	$V_{CC} = 6.3V, V_{OL} = 1.0V$	22		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	15		mA
CKI (Single-pin RC oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−140	−800	μA
All Outputs (I_{OH})	$V_{CC} = 6.3V, V_{OH} = 2.0V$	−75	−480	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−30	−250	μA
Push-Pull Configuration	$V_{CC} = 9.5V, V_{OH} = 4.75V$	−1.4		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 6.3V, V_{OH} = 2.4V$	−1.4		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7				
Outputs, Low Current	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−1.5	−18	mA
Drivers Option (I_{OH})	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.5	−13	mA
LED Configuration, L_0–L_7				
Outputs, High Current	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−3.0	−35	mA
Driver Option (I_{OH})	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−3.0	−25	mA
TRI-STATE Configuration,	$V_{CC} = 9.5V, V_{OH} = 5.5V$	−0.75		mA
L_0–L_7 Outputs, Low	$V_{CC} = 6.3V, V_{OH} = 3.2V$	−0.8		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration,	$V_{CC} = 9.5V, V_{OH} = 5.5V$	−1.5		mA
L_0–L_7 Outputs, High	$V_{CC} = 6.3V, V_{OH} = 3.2V$	−1.6		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−140	μA
CKO Output				
RAM Power Supply Option				
Power Requirement	$V_R = 3.3V$		3.0	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

COP344L/COP345L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	0.75 Watt at 25°C 0.25 Watt at 85°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Optional Operating Voltage (V_{CC})		4.5	7.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		15	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.2	0.3	V
Logic Low (V_{IL})		−0.3		
Schmitt Trigger Input				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)		2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)				
Logic High	I_{OH} = −10 μA	V_{CC}−1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

COP344L/COP345L (Continued)

DC Electrical Characteristics

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted. (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 7.5V, V_{OL} = 0.4V$	1.4		mA
	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0–L_7 Outputs, and Standard	$V_{CC} = 7.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3, D_0–D_3 Outputs (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 7.5V, V_{OL} = 1.0V$	12		mA
High Current Options (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 1.0V$	9		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 7.5V, V_{OL} = 1.0V$	24		mA
Very High Current Options (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 1.0V$	18		mA
	$V_{CC} = 4.5V, V_{OL} = 1.0V$	14		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−100	−900	μA
All Outputs (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	μA
Push-Pull Configuration	$V_{CC} = 7.5V, V_{OH} = 3.75V$	−0.85		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−1.4	−27	mA
Outputs, Low Current	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.4	−17	mA
Driver Option(I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−0.7	−15	mA
LED Configuration, L_0–L_7	$V_{CC} = 7.5V, V_{OH} = 2.0V$	−2.7	−54	mA
Outputs, High Current	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−2.7	−34	mA
Driver Option (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.4	−30	mA
TRI-STATE Configuration,	$V_{CC} = 7.5V, V_{OH} = 4.0V$	−0.7		mA
L_0–L_7 Outputs, Low	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−0.6		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration,	$V_{CC} = 7.5V, V_{OH} = 4.0V$	−1.4		mA
L_0–L_7 Outputs, High	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−1.2		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−200	μA
CKO Output				
RAM Power Supply Option	$V_R = 3.3V$		4.0	mA
Power Requirement				
TRI-STATE Output Leakage Current		−5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics

COP444L/445L: 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 9.5V unless otherwise noted.

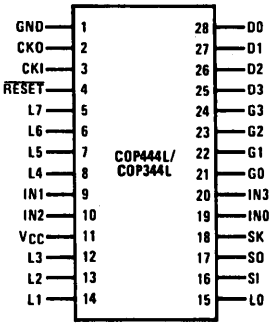
COP344L/345L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time—t_C		16	40	μs
CKI				
Input Frequency—f_I	÷32 Mode	0.8	2.0	MHz
	÷16 Mode	0.4	1.0	MHz
	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 2 MHz		120	ns
Fall Time			80	ns
CKI Using RC (÷4)	R = 56 kΩ ±5% C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
INPUTS:				
IN_3–IN_0, G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

Note 1: Variation due to the device included.

Connection Diagrams

Dual-In-Line

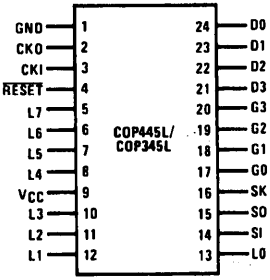


TL/DD/6928-2

Top View

Order Number COP444L-XXX/N or COP344L-XXX/N
See NS Package Number N28B

Dual-In-Line



TL/DD/6928-3

Top View

Order Number COP445L-XXX/N or COP345L-XXX/N
See NS Package Number N24A

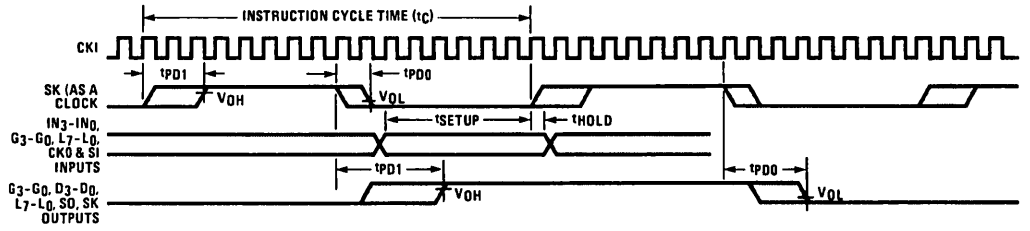
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs (COP444L only)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)

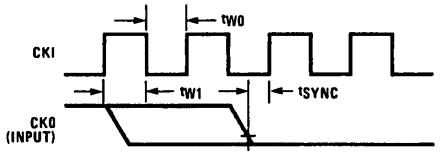
Pin	Description
CKI	System oscillator input
CKO	System oscillator output (or general purpose input, RAM power supply, or SYNC input)
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams



TL/DD/6928-4

FIGURE 3a. Input/Output Timing Diagrams (Crystal Divide-by-16 Mode)



TL/DD/6928-5

FIGURE 3b. Synchronization Timing

Functional Description

A block diagram of the COP444L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

All functional references to the COP444L/COP445L also apply to the COP344L/COP345L.

PROGRAM MEMORY

Program Memory consists of a 2048 byte ROM. As can be seen by an examination of the COP444L/445L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by a 11-bit PC register. Its binary value selects one of the 2048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value. Three levels of subroutine nesting are implemented by the 11-bit subroutine save registers, SA, SB, and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register descriptor, below.)

Four general-purpose inputs, IN_3–IN_0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides a summary of the modes associated with EN_3 and EN_0.

Functional Description (Continued)

Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:
   1. EN_1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
   3. A currently executing instruction has been completed
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

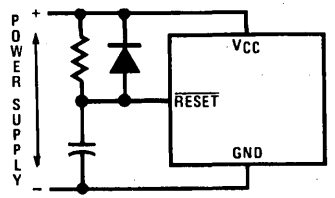
c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user use provide an external RC network and diode to the $\overline{RESET}$ pin as shown below. If the RC network is not used, the $\overline{RESET}$ pin must be pulled up to V_{CC} either by the internal load or by an external resistor (≥40 kΩ) to V_{CC}. The $\overline{RESET}$ pin is configured as a Schmitt trigger input. Initialization will occur whenever a logic "0" is applied to the $\overline{RESET}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/6928–6

RC ≥ 5 x Power Supply Rise Time (R ≥ 40k)

Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instuction at address 0 must be a CLRA.

OSCILLATOR

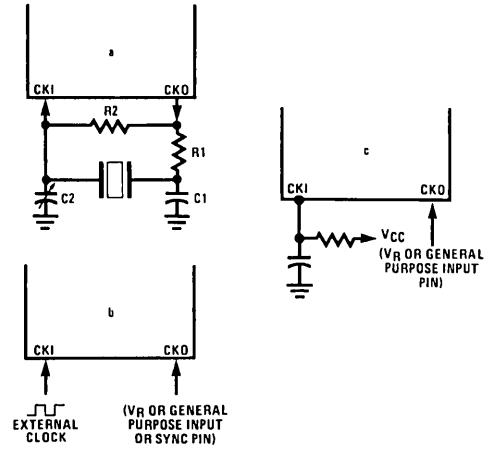
There are four basic clock oscillator configurations available as shown by *Figure 4*.

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32 (optional by 16 or 8).

b. External Oscillator. CKI is an external clock input signal. The external frequency is divided by 32 (optional by 16 or 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R), as a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or as a general purpose input.

Functional Description (Continued)



TL/DD/6928–7

Crystal Oscillator

Crystal Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220
2.097 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (μs)
51	100	19 ± 15%
82	56	19 ± 13%

NOTE: 200 kΩ ≥ R ≥ 25 kΩ
360 pF ≥ C ≥ 50 pF

FIGURE 4. COP444L/445L Oscillator

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP444L/445L system timing configuration does not require use of the CKO pin.

I/O OPTIONS

COP444L/445L outputs have the following optional configurations, illustrated in *Figure 5*.

a. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. Push-Pull—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. Standard L—same as **a.**, but may be disabled. Available on L outputs only.

e. Open Drain L—same as **b.**, but may be disabled. Available on L outputs only.

f. LED Direct Drive—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note: Series current limiting resistors have to be used if the higher operating voltage option is selected and LEDs are driven directly.

g. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

COP444L/COP445L inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT} curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured in **d.**, **e.**, **f.** or **g.**

An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to logic "1".

RAM KEEP-ALIVE OPTION

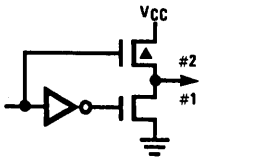
Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the lower four (Br = 0, 1, 2, 3) registers of RAM. To insure that RAM data integrity is maintained, the following conditions *must* be met:

1. $\overline{RESET}$ must go low before V_{CC} goes low during power off; V_{CC} must go high before $\overline{RESET}$ goes high on power-up.
2. V_R must be within the operating range of the chip, and equal to V_{CC} ±1V during normal operation.
3. V_R must be ≥ 3.3V with V_{CC} off.

Functional Description (Continued)

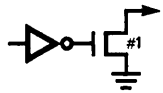
COP445L

If the COP444L is bonded as a 24-pin device, it becomes the COP455L, illustrated in *Figure 2*, COP444L/445L Connection Diagrams. Note that the COP445L does not contain the four general purpose IN inputs (IN_3–IN_0). Use of this option precludes, of course, use of the IN options and the interrupt feature, which uses IN_1. All other options are available for the COP445L.



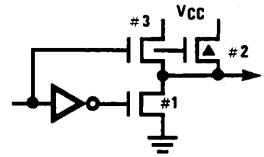
TL/DD/6928–9

a. Standard Output



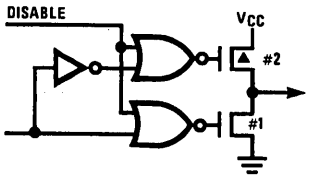
TL/DD/6928–10

b. Open-Drain Output



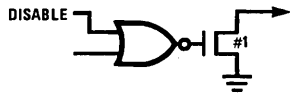
TL/DD/6928–11

c. Push-Pull Output



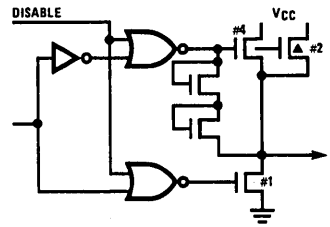
TL/DD/6928–12

d. Standard L Output



TL/DD/6928–13

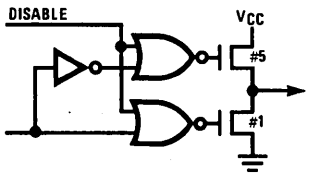
e. Open-Drain L Output



TL/DD/6928–14

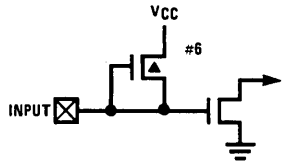
(▲ is Depletion Device)

f. LED (L Output)



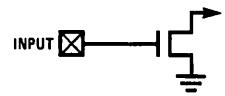
TL/DD/6928–15

g. TRI-STATE Push-Pull (L Output)



TL/DD/6928–16

h. Input with Load



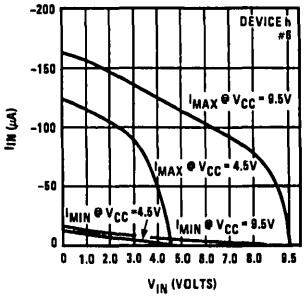
TL/DD/6928–17

I. HI-Z Input

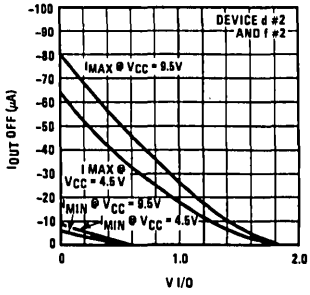
FIGURE 5. Output Configuration

Typical Performance Characteristics

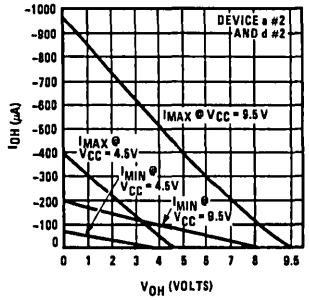
Current for Inputs with Load Device



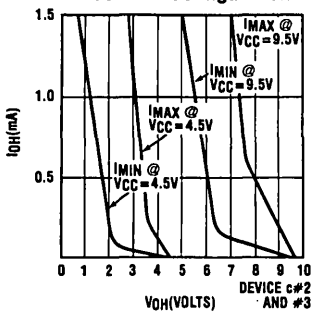
Input Current for L_0 through L_7 when Output Programmed Off by Software



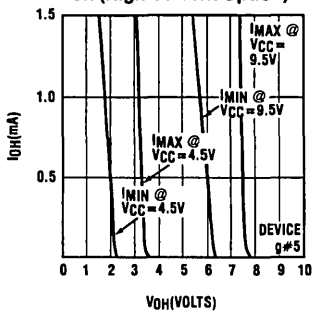
Source Current for Standard Output Configuration



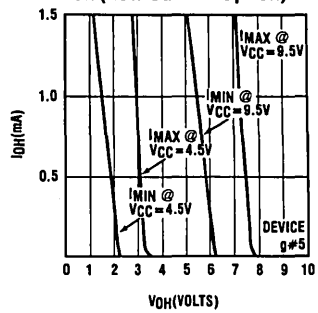
Source Current for SO and SK in Push-Pull Configuration



Source Current for L_0 through L_7 in TRI-STATE Configuration (High Current Option)



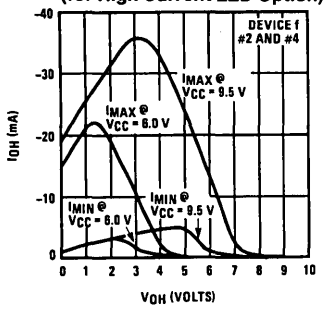
Source Current for L_0 through L_7 in TRI-STATE configuration (Low Current Option)



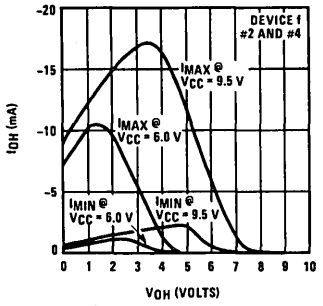
TL/DD/6928–18

Typical Performance Characteristics (Continued)

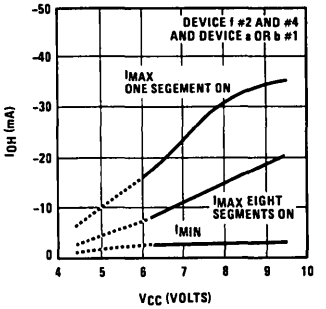
LED Output Source Current (for High Current LED Option)



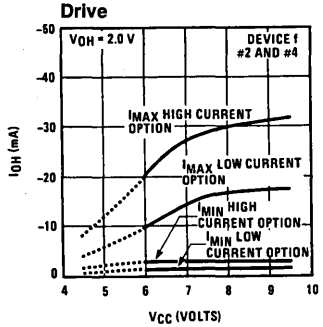
LED Output Source Current (for Low Current LED Option)



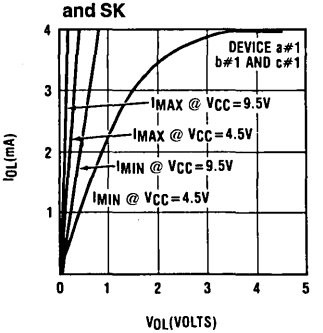
LED Output Direct Segment Drive High Current Options on L_0–L_7 Very High Current Options on D_0–D_3 or G_0–G_3



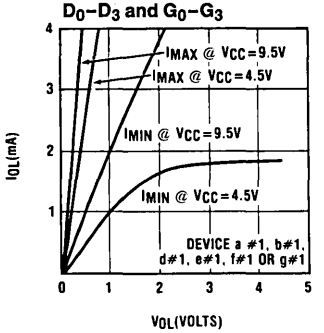
LED Output Direct Segment Drive



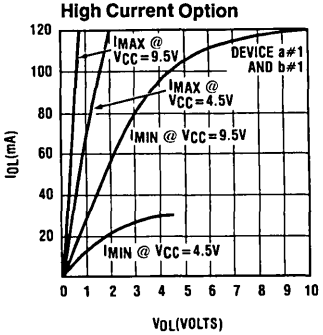
Output Sink Current for SO and SK



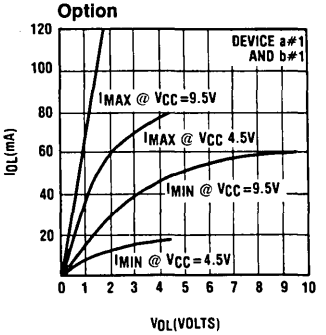
Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3



Output Sink Current G_0–G_3 and D_0–D_3 with Very High Current Option



Output Sink Current for G_0–G_3 and D_0–D_3 (for High Current Option

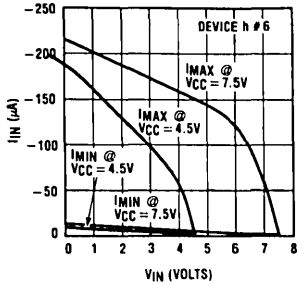


TL/DD/6928–19

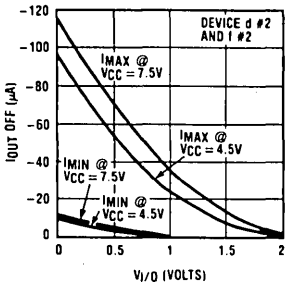
FIGURE 6a. COP444L/COP445L Input/Output Characteristics

Typical Performance Characteristics (Continued)

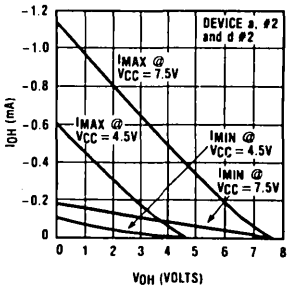
Input Current IN_0–IN_3



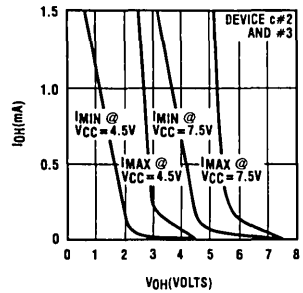
Input Current for L0–L7 when Output Programmed Off by Software



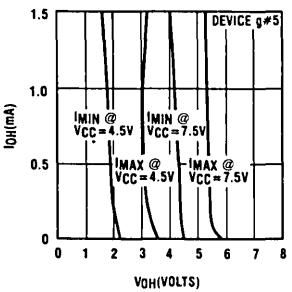
Source Current for Standard Output Configuration



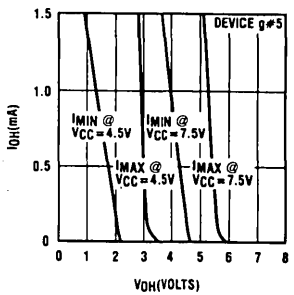
Source Current for SO and SK in Push-Pull Configuration



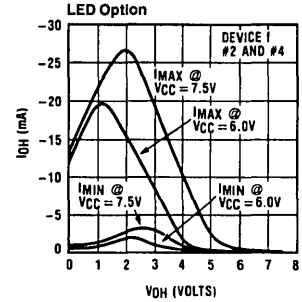
Source Current for L0–L7 in TRI-STATE Configuration (High Current Option)



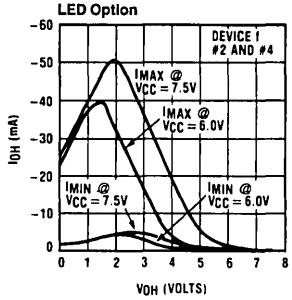
Source Current for L0–L7 in TRI-STATE Configuration (Low Current Option)



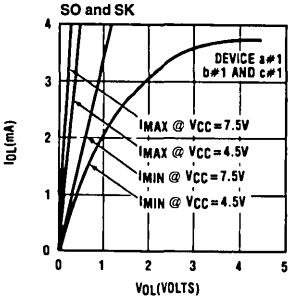
LED Output Source Current (for Low Current LED Option



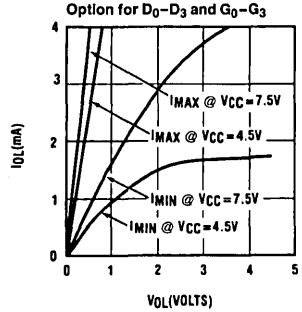
LED Output Source Current (for High Current LED Option



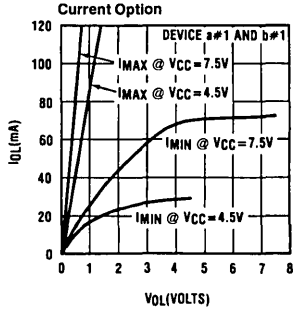
Output Sink Current for SO and SK



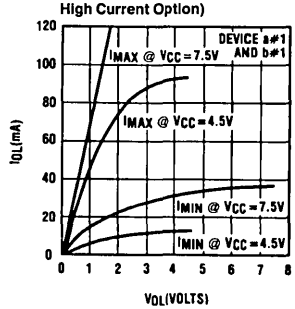
Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3



Output Sink Current G_0–G_3 and D_0–D_3 with Very High Current Option



Output Sink Current for G_0–G_3 and D_0–D_3 (for High Current Option)



TL/DD/6928–20

Fiare 6b. COP344L/COP345L Input/Output Characteristics

COP444L/COP445L/COP344L/COP345L Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP444L/445L instruction set.

TABLE I. COP444L/445L/344L/345L Instruction Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	11-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	3-bit Operand Field, 0-7 binary (RAM Register Select)
a	11-bit Operand Field, 0-2047 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE II. COP444L/445L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	A + 10_{10} → A	None	Add Ten to A
AISC	y	5−	|0101| y |	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
JID		FF	1111 1111	ROM ($PC_{10:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– – –	0110 0 $a_{10:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	– –	1 $a_{6:0}$ (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 4)
		– –	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	0110 1 $a_{10:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101 (r = 0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LDD	r,d	23 – –	0010 0011 0 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 1101 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110 (r = 0:3)	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 – –	0010 0011 1 r d	RAM(r,d) ⟷ A	None	Exchange A with RAM pointed to directly by r,d

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	–7	|00|r|0111| (r = 0:3)	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	|00|r|0100| (r = 0:3)	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	– –	|00|r|(d - 1)| (r = 0:3; d = 0, 9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
			or			
		33	|0011|0011|			
		– –	|1|r|d| any r, any d)			
LEI	y	33	|0001|0011|	y → EN	None	Load EN Immediate (Note 7)
		6–	|0110|y|			
XABR		12	|0001|0010|	A ⟷ Br (0 → A_3)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	|0011|0011|		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
		21	|0010|0001|			
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	G_0 = 0	
	1	11	|0001|0001|	2nd byte	G_1 = 0	
	2	03	|0000|0011|	2nd byte	G_2 = 0	
	3	13	|0001|0011|	2nd byte	G_3 = 0	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1$ = 0	
	2	03	|0000|0011|		$RAM(B)_2$ = 0	
	3	13	|0001|0011|		$RAM(B)_3$ = 0	
SKT		41	|0100|0001|		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)
INPUT/OUTPUT INSTRUCTIONS						
ING		33	|0011|0011|	G → A	None	Input G Ports to A
		2A	|0010|1010|			
ININ		33	|0011|0011|	IN → A	None	Input IN Inputs to A (Note 2)
		28	|0010|1000|			
INIL		33	|0011|0011|	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
		29A	|0010|1001|			

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS (Continued)						
INL		33	0011 | 0011	$L_{7:4} \rightarrow$ RAM(B)	None	Input L Ports to RAM, A
		2E	0010 | 1110	$L_{3:0} \rightarrow$ A		
OBD		33	0011 | 0011	Bd $\rightarrow$ D	None	Output Bd to D Outputs
		3E	0011 | 1110			
OGI	y	33	0011 | 0011	y $\rightarrow$ G	None	Output to G Ports Immediate
		5–	0101 | t			
OMG		33	0011 | 0011	RAM(B) $\rightarrow$ G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A $\leftrightarrow$ SIO, C $\rightarrow$ SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin COP445L or COP345L since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14 or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP444L/445L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, $PC_{10:8}$, A, M. PC_{10}, PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

INIL INSTRUCTION

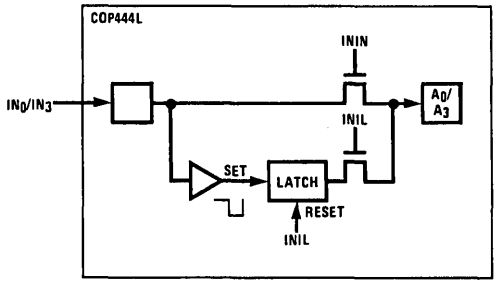
INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 7*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred or the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset; IL_3–IL_0 *not* input on 445L

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC_{10}, PC_9, PC_8 A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 $\rightarrow$ SA $\rightarrow$ SB $\rightarrow$ SC) and replaces the least significant 8 bits of PC as follows: A $\rightarrow$ $PC_{7:4}$, RAM(B) $\rightarrow$ $PC_{3:0}$, leaving PC_{10}, PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC $\rightarrow$ SB $\rightarrow$ SA $\rightarrow$ PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB $\rightarrow$ SC, the previous contents

Description of Selected Instructions (Continued)



TL/DD/6928–21

FIGURE 7. INIL Hardware Implementation

of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP444L/445L to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP444L/445L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last work of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23 or 27 will access data in the next group of four pages.

Option List

The COP444L/445L mask-programmable options are assigned numbers which correspond with the COP*444L* pins.

The following is a list of COP444L options. When specifying a COP445L chip, Options 9, 10, 19, and 20 must all be set to zero. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground Pin—no options available

Option 2: CKO Output
- = 0: clock generator ouput to crystal/resonator (0 not allowable value if option 3 = 3)
- = 1: pin is RAM power supply (V_R) input
- = 2: general purpose input, load device to V_{CC}
- = 3: general purpose input, Hi-Z

Option 3: CKI Input
- = 0: oscillator input divided by 32 (2 MHz max.)
- = 1: oscillator input divided by 16 (1 MHz max.)
- = 2: oscillator input divided by 8 (500 kHz max.)
- = 3: single-pin RC controlled oscillator divided by 4
- = 4: oscillator input divided by 4 (Schmitt)

Option 4: $\overline{\text{RESET}}$ Input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 5: L_7 Driver
- = 0: Standard output
- = 1: Open-drain output
- = 2: High current LED direct segment drive output
- = 3: High current TRI-STATE push-pull output
- = 4: Low-current LED direct segment drive output
- = 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: IN_1 Input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN_2 Input
same as Option 9

Option 11: V_{CC} pin Operating Voltage

	COP44XL	COP34XL
= 0:	+4.5V to +6.3V	+4.5V to +5.5V
= 1:	+4.5V to +9.5V	+4.5V to +7.5V

Option 12: L_3 Driver
same as Option 5

Option 13: L_2 Driver
same as Option 5

Option 14: L_1 Driver
same as Option 5

Option 15: L_0 Driver
same as Option 5

Option 16: SI Input
same as Option 9

Option List (Continued)

Option 17: SO Driver
= 0: standard output
= 1: open-drain output
= 2: push-pull output
Option 18: SK Driver
same as Option 17
Option 19: IN_0 Input
same as Option 9
Option 20: IN_3 Input
same as Option 9
Option 21: G_0 I/O Port
= 0: very-high current standard output
= 1: very-high current open-drain output
= 2: high current standard output
= 3: high current open-drain output
= 4: standard LSTTL output (fanout = 1)
= 5: open-drain LSTTL output (fanout = 1)
Option 22: G_1 I/O Port
same as Option 21
Option 23: G_2 I/O Port
same as Option 21
Option 24: G_3 I/O Port
same as Option 21
Option 25: D_3 Output
same as Option 21
Option 26: D_2 Output
same as Option 21
Option 27: D_1 Output
same as Option 21
Option 28: D_0 Output
same as Option 21
Option 29: L Input Levels
= 0: standard TTL input levels
("0" = 0.8V, "1" = 2.0V)
= 1: higher voltage input levels
("0" = 1.2V, "1" = 3.6V)
Option 30: IN Input Levels
same as Option 29
Option 31: G Input Levels
same as Option 29
Option 32: SI Input Levels
same as Option 29
Option 33: RESET Input
= 0: Schmitt trigger input levels
= 1: standard TTL input levels
= 2: higher voltage input levels
Option 34: CKO Input Levels (CKO=input; Option 2=2, 3)
same as Option 29
Option 35: COP Bonding
= 0: COP444L (28-pin device)
= 1: COP445L (24-pin device)
= 2: both 28- and 24-pin versions
Option 36: Internal Initialization Logic
= 0: normal operation
= 1: no internal initialization logic

COP444L Option Table

The following option information is to be seNt to National along with the EPROM.

OPTION DATA		OPTION DATA	
OPTION 1 VALUE= _____	IS: GROUND PIN	OPTION 21 VALUE= _____	IS: G0 I/O PORT
OPTION 2 VALUE= _____	IS: CKO PIN	OPTION 22 VALUE= _____	IS: G1 I/O PORT
OPTION 3 VALUE= _____	IS: CKI PIN	OPTION 23 VALUE= _____	IS: G2 I/O PORT
OPTION 4 VALUE= _____	IS: RESET INPUT	OPTION 24 VALUE= _____	IS: G3 I/O PORT
OPTION 5 VALUE= _____	IS: L(7) DRIVER	OPTION 25 VALUE= _____	IS: D3 OUTPUT
OPTION 6 VALUE= _____	IS: L(6) DRIVER	OPTION 26 VALUE= _____	IS: D2 OUTPUT
OPTION 7 VALUE= _____	IS: L(5) DRIVER	OPTION 27 VALUE= _____	IS: D1 OUTPUT
OPTION 8 VALUE= _____	IS: L(4) DRIVER	OPTION 28 VALUE= _____	IS: D0 OUTPUT
OPTION 9 VALUE= _____	IS: IN1 INPUT	OPTION 29 VALUE= _____	IS: L INPUT LEVELS
OPTION 10 VALUE= _____	IS: IN2 INPUT	OPTION 30 VALUE= _____	IS: IN INPUT LEVELS
OPTION 11 VALUE= _____	IS: VCC PIN	OPTION 31 VALUE= _____	IS: G INPUT LEVELS
OPTION 12 VALUE= _____	IS: L(3) DRIVER	OPTION 32 VALUE= _____	IS: SI INPUT LEVELS
OPTION 13 VALUE= _____	IS: L(2) DRIVER	OPTION 33 VALUE= _____	IS: RESET INPUT
OPTION 14 VALUE= _____	IS: L(1) DRIVER	OPTION 34 VALUE= _____	IS: CKO INPUT LEVELS
OPTION 15 VALUE= _____	IS: L(0) DRIVER	OPTION 35 VALUE= _____	IS: COP BONDING
OPTION 16 VALUE= _____	IS: SI INPUT	OPTION 36 VALUE= _____	IS: INTERNAL INITIALIZATION LOGIC
OPTION 17 VALUE= _____	IS: SO DRIVER		
OPTION 18 VALUE= _____	IS: SK DRIVER		
OPTION 19 VALUE= _____	IS: IN0 INPUT		
OPTION 20 VALUE= _____	IS: IN3 INPUT		

Typical Applications

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP444L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATION # 1: COP444L GENERAL CONTROLLER

Figure 8 shows an interconnect diagram for a COP444L used as a general controller. Operation of the system is as follows:

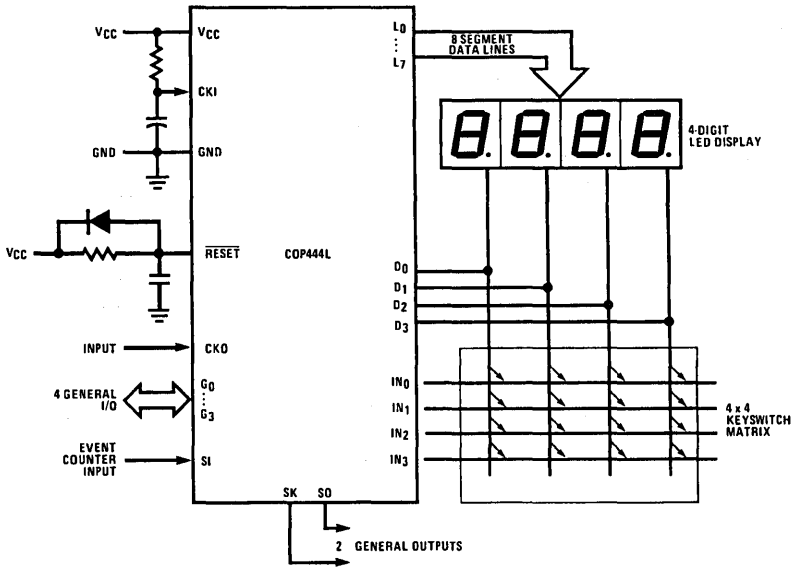
1. The L_7–L_0 outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display
2. The D_3–D_0 outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN_3–IN_0 inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a general-purpose input.
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G_3–G_0) are available for use as required by the user's application.
7. Normal reset operation is selected.

COP444L EVALUATION (See COP Note 4)

The 444L-EVAL is a pre-programmed COP444L, containing several routines which facilitate user familiarization and evaluation of the COP444L operating characteristics. It may be used as an up/down counter or timer, interfacing to any combination of (1) an LED digit or lamps, (2) 4-digit LED Display Controller, (3) a 4-digit VF Display Controller, and/or (4) a 4-digit LCD Display Controller. Alternatively, it may be used as a simple music synthesizer.

SAMPLE CIRCUITS

1. By making only the oscillator, power supply and "L7" connections, *(Figure 9)* an approximate 1 Hz square wave will be produced at output "D1." This output may be observed with an oscilloscope, or connected to additional TTL or CMOS circuitry.
2. By making the indicated connections to a small LED digit (NSA1541A, NSA1166, or equiv.—larger digits will be proportionately dimmer), the counter actions may be observed. Place the "up/down" switch in the "up" (open) position and apply a TTL-compatible signal at the "counter-input." Placing the "up/down" switch in the "down" (closed) position causes the count to decrement on each high-to-low input transition.
3. All 4 digits of the counter may be displayed by connecting a standard display controller (COP470 for VF, COP472 for LCD, MM5450 for LED) as shown in *Figure 9*.



TL/DD/6928–22

FIGURE 8. COP444L Keyboard/Display Interface

Typical Applications (Continued)

Any combination of the single LED digit and display controllers may be used simultaneously, and will display the same data.

4. The simple counter described above becomes a timer when the 1 Hz output is connected to the "counter input." Up or down counting may be used with input frequencies up to 1 kHz. Improved timing accuracies may be obtained by subsituting the 2.097 MHz crystal oscillator circuit of *Figure 4a* for the RC network shown in *Figure 9*, or by connecting a more stable external frequency to the "counter input" in place of the 1 Hz signal.
5. An "entertaining" use of the 444L-EVAL is as a simple music synthesizer (or electronic organ). By attaching a simple switch matrix (or keyboard), a speaker or piezo-ceramic transducer, and grounding "L7", the user can play "music" *(Figure 10)*. Three modes of operation are available: Play a note, play one of four stored tunes, or record a tune for subsequent replay.

   a. *Play A Note*

   Twelve keys, representing the 12 notes in one octave, are labeled "C" through "B"; depressing a key causes a square wave of the corresponding frequency to be outputted to the speaker. Depressing "LShift" or "UShift" causes the next note to be shifted to the next lower octave (one-half frequency) or the next upper octave (double frequency), respectively.

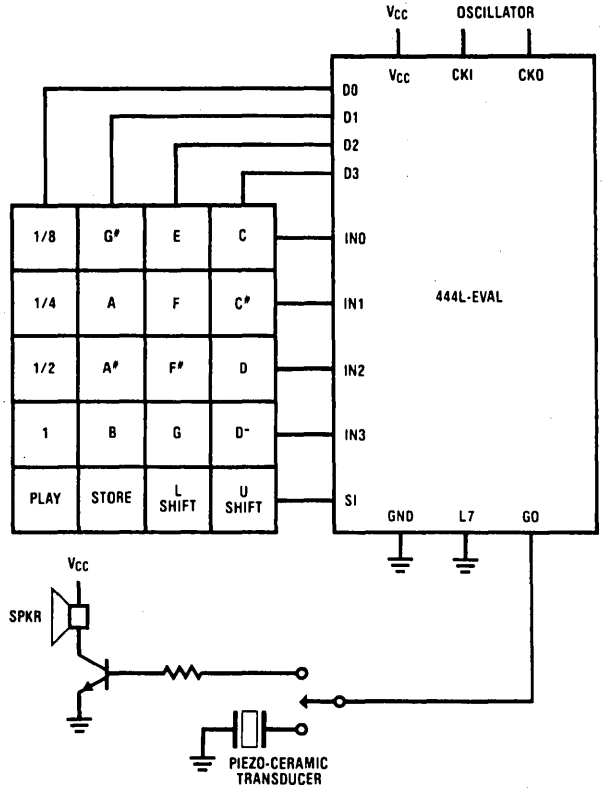
   b. *Play Stored Tune*

   Depressing "Play" followed by "⅛", "¼", "½", or "1" will cause one of 4 stored tunes to be played.

   c. *Record Tune*

   Any combination of notes and rests up to a total of 48 may be stored in RAM for later replay. To store a note, press the appropriate note key, followed by the duration of the note (⅛-note, ¼-note, ½-note, whole (1)-note, followed by "Store"; a rest is stored by selecting the duration and pressing "Store." When the tune is complete, press "Play" followed by "Store"; the tune will be played for immediate audition. Subsequent depression of "Play" and "Store" will replay the last stored tune.

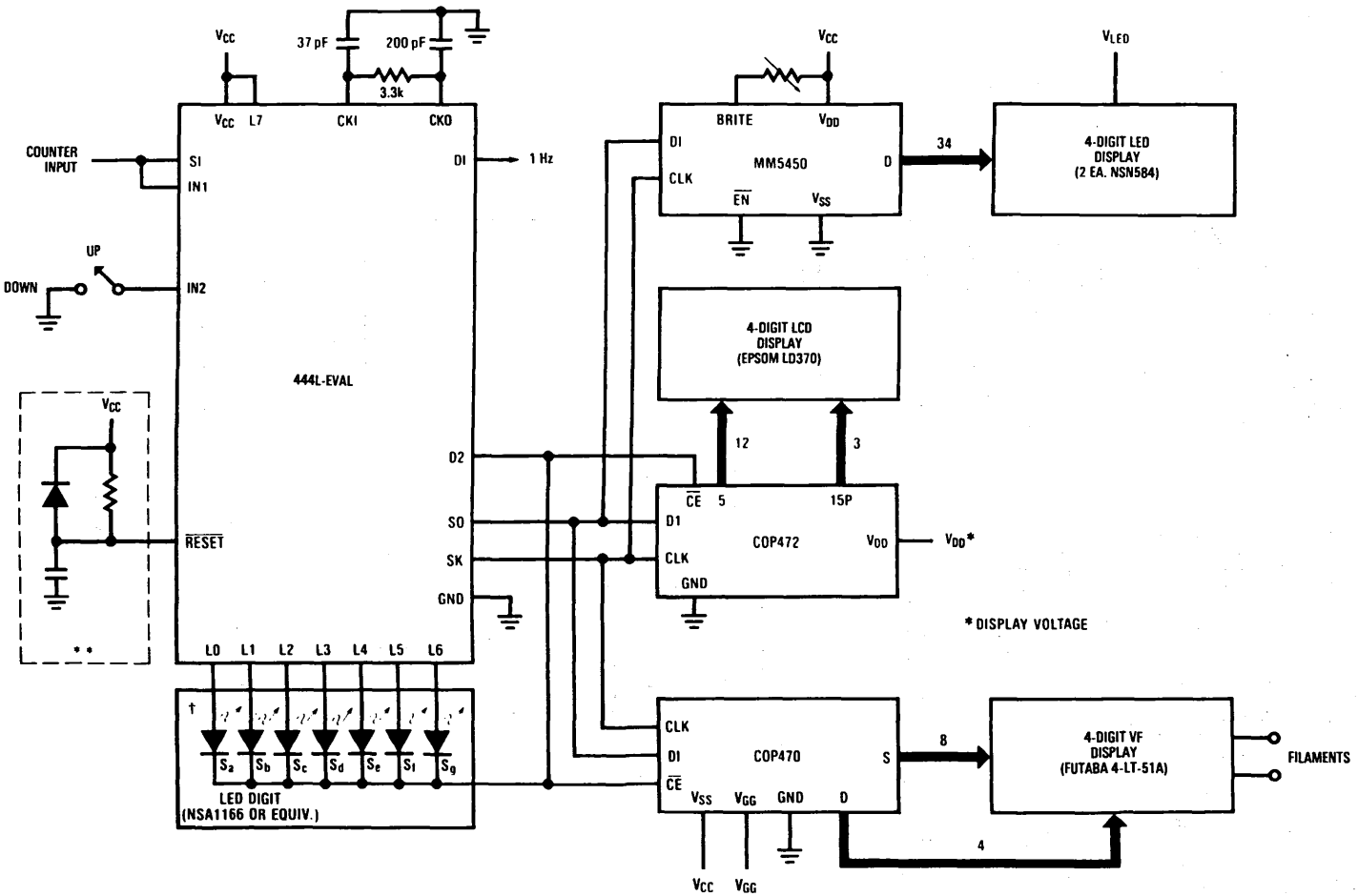
Note: The accuracy of the tones produced is a function of the oscillator accuracy and stability; the crystal oscillator is recommended.



TL/DD/6928-23

FIGURE 9. Counter/Timer

Typical Applications (Continued)



TL/DD/6928–24

†If LED Direct Drive option is selected with power supply operating voltage of 4.5V to 9.5V (higher voltage option), series resistors must be used to limit current.

** See "Initialization"

FIGURE 10. Music Synthesizer

National Semiconductor

COP401L ROMless N-Channel Microcontroller

General Description

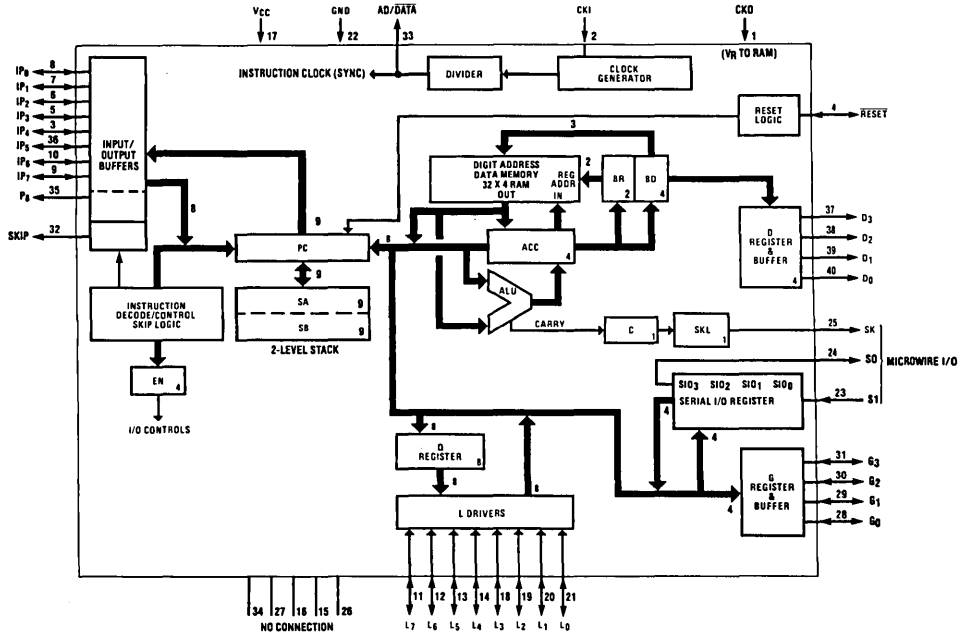
The COP401L ROMless Microcontroller is a member of the COPS™ family of microcontrollers, fabricated using N-channel, silicon gate MOS technology. The COP401L contains CPU, RAM, I/O and is identical to a COP410L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP401L will perform exactly as the COP410L. This important benefit facilitates development and debug of a COP program prior to masking the final part.

The COP401L is intended for emulation only, not intended for volume production. Use COP402 or COP404L for volume production.

Features

- Circuit equivalent of COP410L
- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- Separate RAM power supply pin for RAM keep-alive applications
- Two-level subroutine stack
- 15 μs instruction time
- Single supply operation (4.5–9.5V)
- Low current drain (8 mA max.)
- Internal binary counter register with serial I/O
- MICROWIRE™ compatible serial I/O
- General purpose outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Pin-for-pin compatible with COP402 and COP404L

Block Diagram



TL/DD/6913–1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C
	0.4W at 70°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 2)	4.5	9.5	V
Power Supply Ripple	Peal to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})		2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
IP0–IP7 Input Levels				
Logic High	V_{CC} = 9.5V	2.4		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
All Other Inputs				
Logic High	V_{CC} = 9.5V	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 µA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
IP0–IP7, P8, SKIP	(Note 1)			
Logic Low	I_{OL} = 1.6 mA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{CC} = 9.5V, V_{OL} = 0.4V	1.8		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L_0–L_7 and G_0–G_3 Outputs	V_{CC} = 9.5V, V_{OL} = 0.4V	0.8		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
D_0–D_3 Outputs	V_{CC} = 9.5V, V_{OL} = 1.0V	30		mA
	V_{CC} = 4.5V, V_{OL} = 1.0V	15		mA
CKO				
RAM Power Supply Input	V_R = 3.3V		1.5	mA

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Source Current				
D_0–D_3, G_0–G_3 Outputs (I_{OH})	$V_{CC} = 9.5V$, $V_{OH} = 2.0V$	−140	−800	μA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−30	−250	μA
SO and SK Outputs (I_{OH})	$V_{CC} = 9.5V$, $V_{OH} = 4.75V$	−1.4		mA
	$V_{CC} = 4.5V$, $V_{OH} = 1.0V$	−1.2		mA
L_0–L_7 Outputs	$V_{CC} = 9.5V$, $V_{OH} = 2.0V$	−3.0	−35	mA
	$V_{CC} = 6.0V$, $V_{OH} = 2.0V$	−0.3	−25	mA
Input Load Source Current (I_{IL})	$V_{CC} = 5.0V$, $V_L = 0V$	−10	−140	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
All Other Pins			1.5	mA

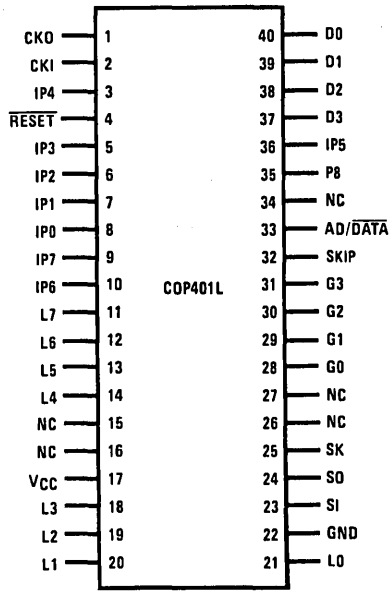
AC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		15	40	μs
CKI				
Input Frequency f_I	(÷32 Mode)	0.8	2.1	MHz
Duty Cycle		30	60	%
Rise Time			120	ns
Fall Time	$f_I = 2.097$ MHz		80	ns
INPUTS:				
SI, IP7–IP0				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = pF, $V_{OUT} = 1.5V$			
SO, SK Outputs	$R_L = 20\ k\Omega$			
t_{pd1}, t_{pd0}			4.0	μs
D_3–D_0, G_3–G_0, L_7–L_0	$R_L = k\Omega$			
t_{pd1}, t_{pd0}			5.6	μs
IP7–IP0, P8, SKIP	$R_L = 5\ k\Omega$			
t_{pd1}, t_{pd0}			7.2	μs

Note 1: Pull-up resistors required.

Note 2: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Connection Diagram



TL/DD/6913-2

Order Number COP401L/N
NS Package Number N40A

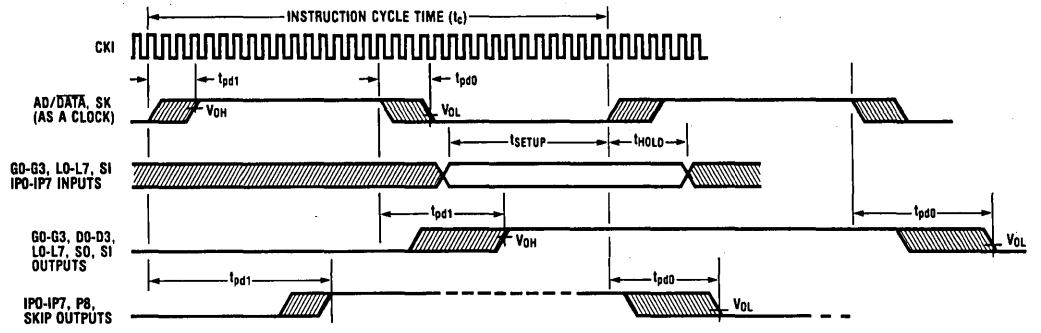
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with LED segment drive
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{\text{DATA}}$	Address Out/data in flag

Pin	Description
CKI	System oscillator input
CKO	RAM power supply input
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8	Most significant ROM address bit output
SKIP	Instruction skip output

Timing Diagram



TL/DD/6913-3

FIGURE 3. Input/Output

Functional Description

A block diagram of the COP401L is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

PROGRAM MEMORY

Program Memory consists of a 512-byte external memory. As can be seen by an examination of the COP401L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

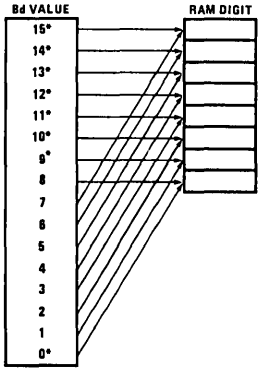
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table 3).



TL/DD/6913–4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP401L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift register.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP401L operation.

Functional Description (Continued)

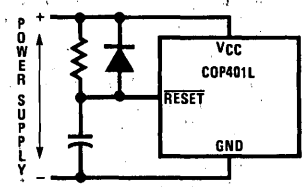
TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below *(Figure 5)*. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/6913–5

RC ≥ Power Supply Rise Time

FIGURE 5. Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP401L is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE® outputs
3. TTL-compatible inputs
4. access time = 5 μs max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P8 and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P8 is a dedicated address output, and does not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

CKI is an external clock input signal. The external frequency is divided by 32 to give the instruction cycle time. The divide-by-32 configuration was chosen to make the COP 401L compatible with the COP404L and the COPS™ Development System. However, the ÷32 configuration is not available on the COP410L/COP411L. It is therefore possible to exactly emulate the system **speed** (cycle time), but **not** possible to drive the 401L with the system clock during emulation.

Functional Description (Continued)

CKO (RAM POWER)

CKO is configured as a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. This pin must be connected to V_{CC} if the power backup feature is not used. To insure that RAM integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. During normal operation, V_R must be within the operating range of the chip with $(V_{CC}-1) \leq V_R \leq V_{CC}$.
3. V_R must be $\geq$ 3.3V with V_{CC} off.

INPUT/OUTPUT CONFIGURATIONS

COP401L outputs have the following configurations, illustrated in *Figure 6*:

a. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. (Used on D and G outputs.)

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. (Used on IP, P and SKIP outputs.)

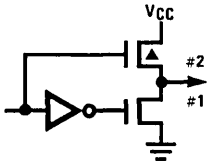
c. Push-Pull—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. (Used on SO and SK outputs.)

d. LED Direct Drive—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. (Used on L outputs.)

COP401L inputs have an on-chip depletion load device to V_{CC}.

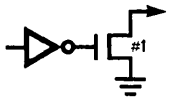
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of five devices (numbered 1–5, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 7* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 7*, Device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to a logic "1".



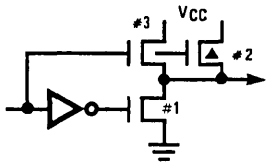
TL/DD/6913-6

a. Standard Output



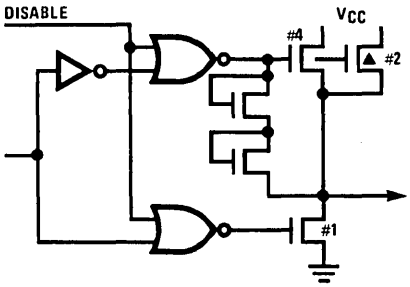
TL/DD/6913-7

b. Open-Drain Output



TL/DD/6913-8

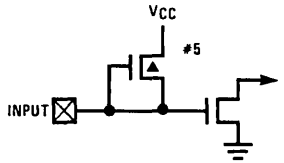
c. Push-Pull Output



TL/DD/6913-9

(▲ is Depletion Device)

d. L Output (LED)



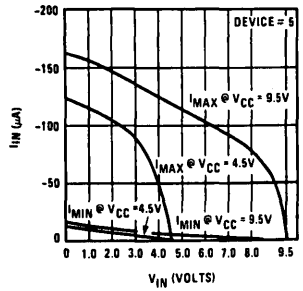
TL/DD/6913-10

e. Input with Load

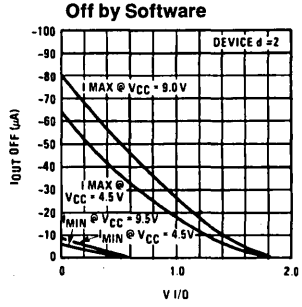
FIGURE 6. Output Configurations

Typical Performance Characteristics

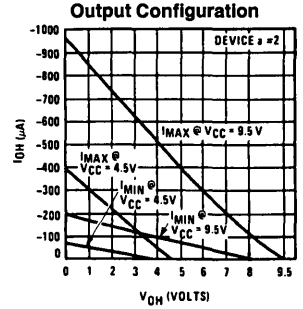
Current for Inputs with Load Device



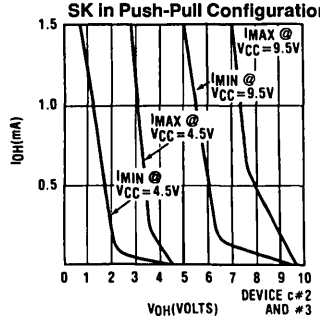
Input Current for L_0 through L_7 when Output Programmed Off by Software



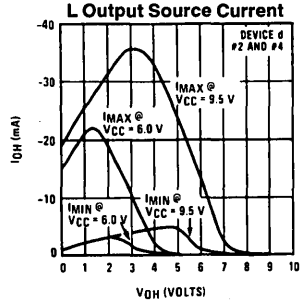
Source Current for Standard Output Configuration



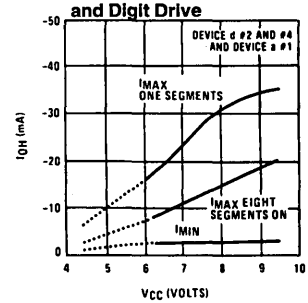
Source Current for SO and SK in Push-Pull Configuration



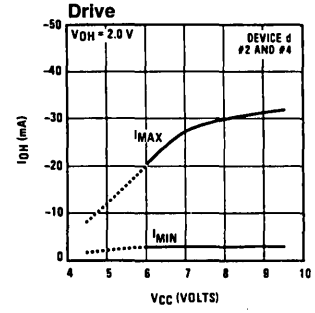
L Output Source Current



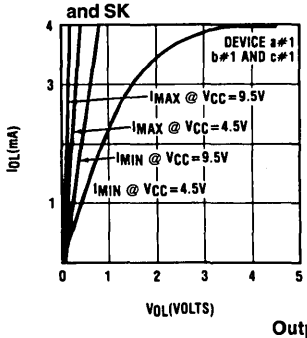
LED Output Direct Segment and Digit Drive



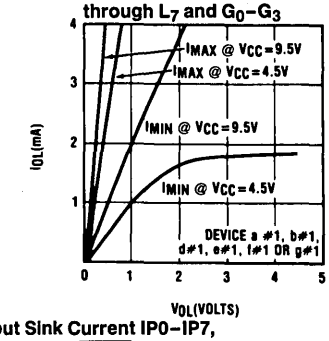
LED Output Direct Segment Drive



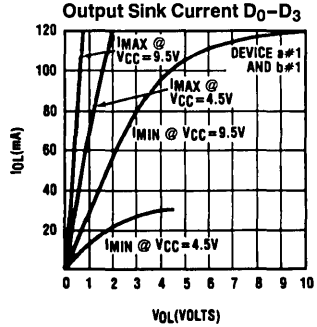
Outut Sink Current for SO and SK



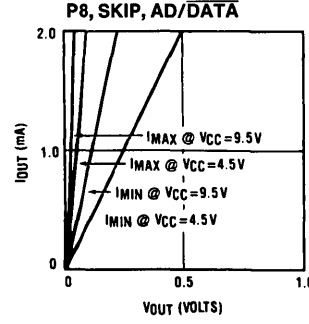
Output Sink Current for L_0 through L_7 and G_0–G_3



Output Sink Current D_0–D_3



Output Sink Current IP0–IP7, P8, SKIP, AD/DATA



TL/DD/69134–11

FIGURE 7. I/O Characteristics

COP401L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP401L instruction set.

TABLE II. COP401L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP401L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5−	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

COP410L Instruction Set (Continued)

TABLE III. COP401L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	0110 | 000 | a_8	a → PC	None	Jump
		– –	$a_{7:0}$			
JP	a	– –	1 | $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 3)
			or			
		– –	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 | $a_{5:0}$	PC + 1 → SA → SB	None	Jump to Subroutine Page (Note 4)
				010 → $PC_{8:6}$ a → $PC_{5:0}$		
JSR	a	6–	0110 | 100 | a_8	PC + 1 → SA → SB	None	Jump to Subroutine
		– –	$a_{7:0}$	a → PC		
RET		48	0100 | 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 | 0011	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 | 1100	RAM(B) → $Q_{3:0}$		
LD	r	–5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8, A, M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23	0010 | 0011	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
		BF	1011 | 1111			
XDS	r	–7	00 | r | 0111	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	00 | r | 0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

COP410L Instruction Set (Continued)

TABLE III. COP401L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r, d	–	00 | r | (d - 1) (d = 0, 9:15)	r, d → B	Skip until not a LBI	Load B Immediate with r, d (Note 5)
LEI	y	33 6–	0011 | 0011 0110 | y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 | 0011 0010 | 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0 = 0$	
	1	11	0001 | 0001		$G_1 = 0$	
	2	03	0000 | 0011		$G_2 = 0$	
	3	13	0001 | 0011		$G_3 = 0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1 = 0$	
	2	03	0000 | 0011		$RAM(B)_2 = 0$	
	3	13	0001 | 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 | 0011 0010 | 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 | 0011 0010 | 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 | 0011 0011 | 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (010 is loaded into the upper 3 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus* 1, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP401L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 $\rightarrow$ SA $\rightarrow$ SB) and replaces the least significant 8 bits of PC as follows: A $\rightarrow$ $PC_{7:4}$, RAM(B) $\rightarrow$ $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB $\rightarrow$ SA $\rightarrow$ PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA $\rightarrow$ SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA $\rightarrow$ SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP401L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or 7 will access data in the next group of 4 pages.

Typical Applications

PROM-BASED SYSTEM

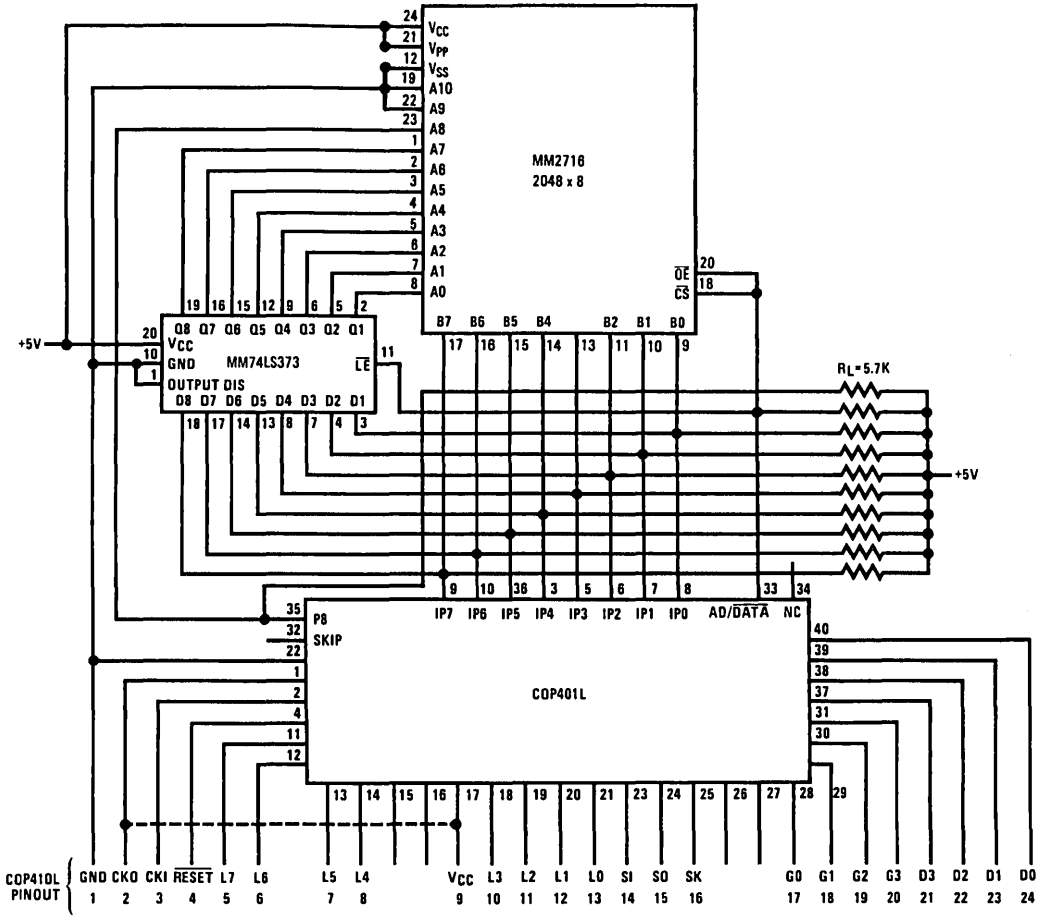
The COP401L may be used to emulate the COP410L. *Figure 8* shows the interconnect to implement a COP401L hardware emulation. This connection uses one MM5204 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP_7–IP_0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP_7–IP_0 output addresses. The 8-bit latch (MM74C373) latches the address to drive the memory.

When AD/$\overline{\text{DATA}}$ turns off, the EPROM is enabled and the IP_7–IP_0 pins will input the memory data. P8 outputs the most significant address bit to the memory. (SKIP output may be used for program debug if needed.)

24 of the COP401L pins may be configured exactly the same as a COP410L.

Typical Applications (Continued)



TL/DD/6913–12

FIGURE 8. COP401L Used to Emulate a COP410L

Option Table

COP401L MASK OPTIONS

The following COP410L options have been implemented in this basic version of the COP401L.

Option Value	Comment
Option 1 = 0	Ground—no option
Option 2 = 1	CKO is RAM power supply input
Option 3 = N/A	CKI is external clock divide-by-32 (not available on COP410L)
Option 4 = 0	Reset has load to V_{CC}
Option 5 = 2	
Option 6 = 2	L outputs are LED direct-drive
Option 7 = 2	
Option 8 = 2	
Option 9 = 1	V_{CC} pin 4.5V to 9.5V operation
Option 10 = 2	
Option 11 = 2	L outputs are LED direct-drive
Option 12 = 2	
Option 13 = 2	

Option Value	Comment
Option 14 = 0	SI has load to V_{CC}
Option 15 = 2	SO is push-pull output
Option 16 = 2	SK is push-pull output
Option 17 = 0	
Option 18 = 0	G outputs are standard
Option 19 = 0	
Option 20 = 0	
Option 21 = 0	
Option 22 = 0	D outputs are standard
Option 23 = 0	very high current
Option 24 = 0	
Option 25 = 0	L
Option 26 = 0	G Have standard TTL input levels
Option 27 = 0	SI
Option 28 = N/A	40-pin package

National Semiconductor

COP401L-X13/COP401L-R13 ROMless N-Channel Microcontroller

General Description

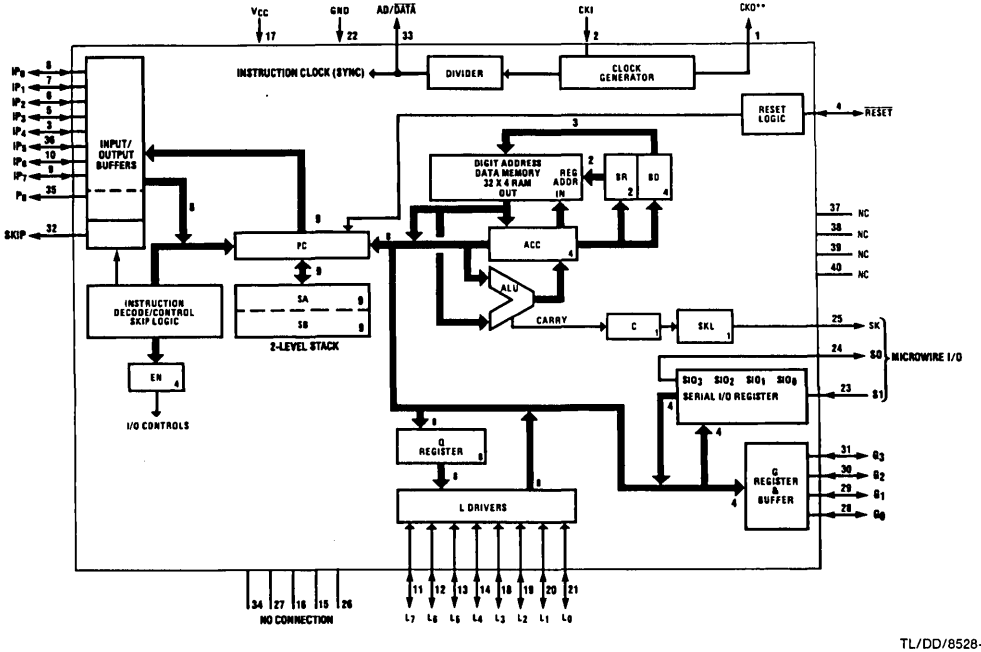
The COP401L-X13/COP401L-R13 ROMless Microcontrollers are members of the COPS™ family of microcontrollers, fabricated using N-channel, silicon gate MOS technology. The COP401L-X13/COP401L-R13 contain CPU, RAM, I/O and are identical to a COP413L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP401L-X13/COP401L-R13 will perform exactly as the COP413L. This important benefit facilitates development and debug of a COP program prior to masking the final part.

There are two clock oscillator configurations available. The crystal oscillator configuration is called COP401L-X13 and the RC oscillator configuration is called COP401L-R13.

Features

- Circuit equivalent of COP413L
- Low cost
- Powerful instruction set
- 512 × 8 ROM, 32 × 4 RAM
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5–5.5V)
- Low current drain (8 mA max)
- Internal binary counter register with serial I/O
- MICROWIRE™ compatible serial I/O
- General purpose outputs
- Software/hardware compatible with other members of COP400 family
- Pin-for-pin compatible with COP402 and COP404L
- High noise immunity inputs (V_{IL} = 1.2V, V_{IH} = 3.6V)

Block Diagram



TL/DD/8528–1

**COP401L-X13 only

FIGURE 1

COP401L-X13/COP401L-R13 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.3 to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation COP413L	0.3 Watt at 70°C
Total Source Current	25 mA
Total Sink Current	40 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})		3.0		V
Logic Low (V_{IL})			0.4	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low			0.6	V
SO Input Level (Test Mode)	(Note 2)	2.5		V
IP0–IP7, SI Input Level				
Logic High	(TTL Level)	2.0		V
Logic Low			0.8	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance			7	pF
Reset Input Leakage		−1	+1	μA
Output Current Levels				
Output Sink Current (I_{OL})				
SO and SK Outputs	V_{OL} = 0.4V	0.9		mA
L0–L7 Outputs, G0–G3	V_{OL} = 0.4V	0.4		mA
CKO	V_{OL} = 0.4V	0.2		mA
IP0–IP7, P8, SKIP, AD/$\overline{DATA}$	V_{OL} = 0.4V	1.6		mA
Output Source Current (I_{OH})				
L0–L7 G0–G3, SO, SK	V_{OH} = 2.4V	−25		μA
IP0–IP7, P8, SKIP, AD/$\overline{DATA}$	V_{OH} = 2.4V	−25		μA
SO, SK	V_{OH} = 1.0V	−1.2		mA
IP0–IP7, P8, SKIP, AD/$\overline{DATA}$	V_{OH} = 1.0V	−1.2		mA
SI Input Load Source Current	V_{IL} = 0V	−10	−140	μA
Total Sink Current Allowed				
L7–L4, G Port			4	mA
L3–L0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

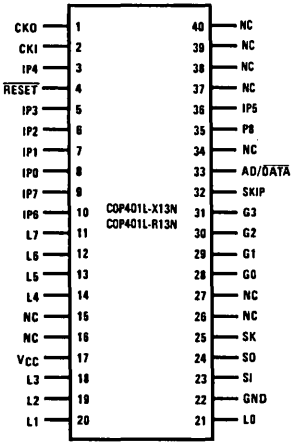
Note 2: SO output "0" level must be less than 0.8V for normal operation.

AC Electrical Characteristics $0°C \le T_A \le 70°C, 4.5V \le V_{CC} \le 5.5V$

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time - t_c		16	40	μs
CKI				
Input Frequency - fi	÷ 8 Mode	0.2	0.5	MHz
Duty Cycle		30	60	%
Rise Time	fi = 0.5 MHz		500	ns
Fall Time			200	ns
CKI Using RC (÷ 4)	R = 56 kΩ ±5% C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
Inputs:				
G3–G0, L7–L0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI, IP0–IP7				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: C_L = 50 pF, V_{OUT} = 1.5V			
SO, SK Outputs	R_L = 20 kΩ			
tpd1, tpd0			4.0	μs
L, G Outputs	R_L = 20 kΩ			
tpd1, tpd0			5.6	μs
IP0–IP7, P8, SKIP	R_L = 5 kΩ			
tpd1, tpd0			7.2	μs

Note 1: Variation due to the device included.

Connection Diagram



TL/DD/8528-2

FIGURE 2

Order Number COP401L-X13N or COP401L-R13N
See NS Package Number N40A

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports
G_3–G_0	4 bidirectional I/O ports
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{DATA}$	Address out/data in flag
CKI	System oscillator input
CKO	System oscillator output or NC
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8	Most significant ROM address bit output
SKIP	Instruction skip output

1

Timing Waveform

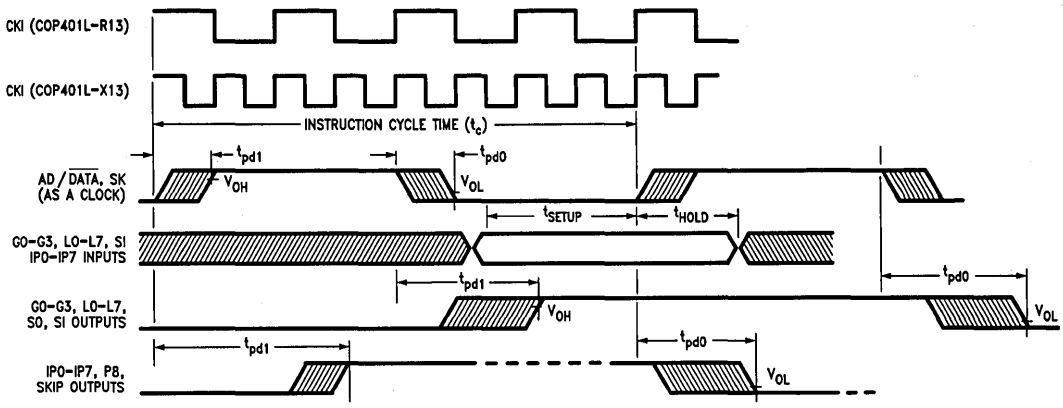


FIGURE 3. Input/Output Timing Diagram

Development Support

The MOLE (Microcontroller On Line Emulator) is a low cost development system and real time emulator for COP's products. They also include TMP, 8050, and the new 16-bit HPC Microcontroller Family. The MOLE provides effective support for the development of both software and hardware in the user's application.

The purpose of the MOLE is to provide a tool to write and assemble code, emulate code for the target microcontroller and assist in debugging of the system.

The MOLE can be connected to various hosts, IBM PC STARPLEX™, Kaypro, Apple, and Intel Systems, via RS-232 port. This link facilitate the up loading/down loading of code, supports host assembly and mass storage.

The MOLE consists of three parts; brain, personality and optional host software.

The brain board is the computing engine of the system. It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, from programming and diagnostic operation. It has three serial ports which can be connected to a terminal, host system, printer, modem or to other MOLE's in a multi-MOLE environment.

The personality board contains the necessary hardware and firmware needed to emulate the target microcontroller. The emulation cable which replaces the target controller attaches to this board. The software contains a cross assembler and communications program for up loading and down loading code from the MOLE.

MOLE Ordering Information

P/N	Description
MOLE-BRAIN	MOLE Computer Board
MOLE-COPS-PB1	COPS' Personality Board
MOLE-XXX-YYY	Optional Software

Where XXX = COPS, TMP, 8050, or HPC

YYY = Host System, IBM, APPLE, KAY (Kaypro), CP/M

Functional Description

A block diagram of the COP401L-X13/COP401L-R13 is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

PROGRAM MEMORY

Program Memory consists of a 512-byte external memory. As can be seen by an examination of the COP401L-X13/ COP401L-R13 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

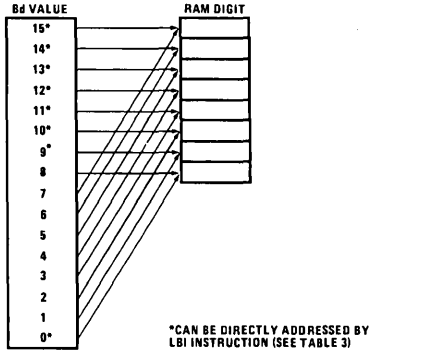
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table 3).



TL/DD/8528–4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP401L-X13/COP401L-R13, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in-/serial-out shift register or as a binary counter depending on the contents of the EN Register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN registers (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO Register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI Input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO Output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data pesent at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP401L-X13/ COP401L-R13 operation.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high impedance input state.

Functional Description (Continued)

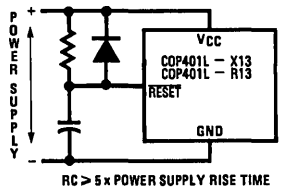
TABLE I. Enable Register Modes - Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1, SK=Clock If SKL=0, SK=0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL=1, SK=Clock If SKL=0, SK=0
0	1	Binary Counter	Input to Binary Counter	0	If SKL=1, SK=1 If SKL=0, SK=0
1	1	Binary Counter	Input to Binary Counter	1	If SKL=1, SK=1 If SKL=0, SK=0

4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table 1 provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below (*Figure 5*). The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/8528-5

Figure 5. Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP401L-X13/COP401L-R13 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE® outputs
3. TTL-compatible inputs
4. access time = 5 μs max.

Typically these requirements are met using bipolar or MOS PROMs.

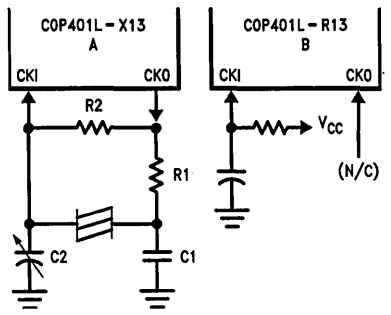
During operation, the address of the next instruction is sent out on P8 and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P8 is a dedicated address output, and does not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

There are two basic clock oscillator configurations available as shown by *Figure 6*.

a. The COP401L-X13 is a Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.

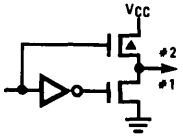
b. The COP401L-R13 is a RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO becomes no connection.



TL/DD/8528-6

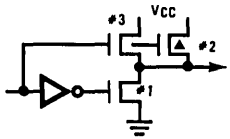
FIGURE 6. COP401L-X13/COP401L-R13 Oscillator

Functional Description (Continued)



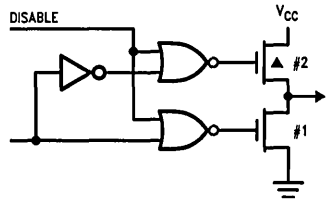
TL/DD/8528-7

a. Standard Output



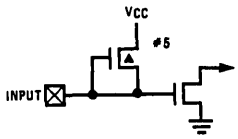
TL/DD/8528-8

b. Push-Pull Output



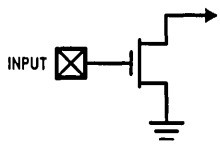
TL/DD/8528-9

c. Standard L Ouput



TL/DD/8528-10

d. Input With Load



TL/DD/8528-11

e. Hi-Z Input

FIGURE 7. Input and Output Configurations

Ceramic Resonator Oscillator

Resonator Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (in μs)
51	100	19 ± 15%
82	56	19 ± 13%

Note: 200 kΩ ≥ R ≥ 25 kΩ
220 pF ≥ C ≥ 50 pF

I/O CONFIGURATIONS

COP401L-X13/COP401L-R13 inputs and outputs have the following configurations, illustrated in *Figure 7*.

a. G0–G3—an enhancement mode device to ground in conjunction with depletion-mode device to V_{CC}.

b. SO, SK, IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$—an enhancement mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

c. L0–L7—same as a, but may be disabled.

d. SI has on-chip depletion load device to V_{CC}.

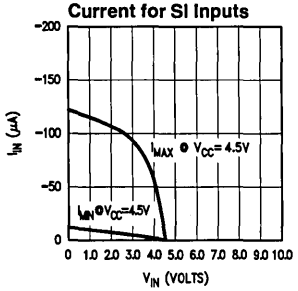
e. $\overline{\text{RESET}}$ has a Hi-Z input which must be driven to a "1" or "0" by external components.

Curves are given in *Figure 8* to allow the designer to effectively use the I/O configurations in designing a system.

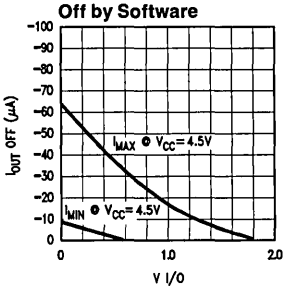
An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current, however, when the L lines are used as inputs, the disabled depletion device can not be relied on to source sufficient current to pull an input to a logic "1".

Typical Performance Characteristics

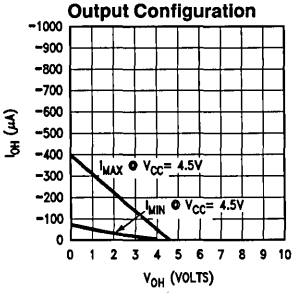
Current for SI Inputs



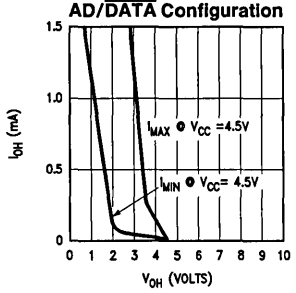
Input Current for L_0 through L_7 when Output Programmed Off by Software



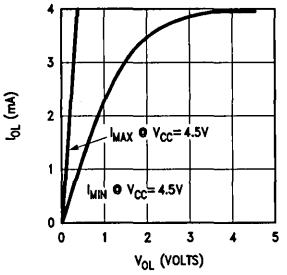
Source Current for L7–L0, G3–G0 Output Configuration



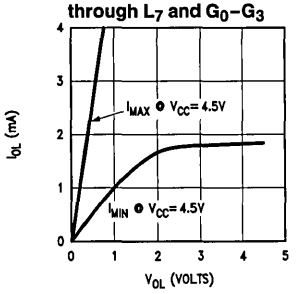
Source Current for SO, SK, IP0, IP7, P8, SKIP, AD/$\overline{\text{DATA}}$ Configuration



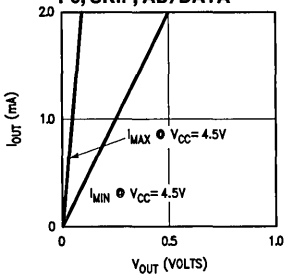
Output Sink Current for SO and SK



Output Sink Current for L_0 through L_7 and G_0–G_3



Output Sink Current IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$



TL/DD/8528–12

FIGURE 8. I/O Characteristics

COP401L-X13/COP401L-R13 Instruction Set

Table II is a symbol table providing internal architecture, Instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP401L-X13/COP401L-R13 instruction set.

TABLE II. COP401L-X13/COP401L-R13 Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic Controlled Clock Output
Instruction Operand Symbols	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
Operational Symbols	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP401L-X13/COP401L-R13 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 | 0000	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 | 0001	A+RAM(B) → A	None	Add RAM to A
AISC	y	5–	0101 | y	A+y → A	Carry	Add Immediate, Skip on Carry (y≠0)
CLRA		00	0000 | 0000	0 → A	None	Clear A
COMP		40	0100 | 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 | 0100	None	None	No Operation
RC		32	0011 | 0010	"0" → C	None	Reset C
SC		22	0010 | 0010	"1" → C	None	Set C
XOR		02	0000 | 0010	A⊕RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM(PC_8,A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	0110 | 000 | a_8	a → PC	None	Jump
		–	$a_{7:0}$			
JP	a	–	1 | $a_{6:0}$ (pages 2, 3 only) or	a → $PC_{6:0}$	None	Jump within-Page (Note 3)
		–	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	10 | $a_{5:0}$	PC+1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6–	0110 | 100 | a_8	PC+1 → SA → SB a → PC	None	Jump to Subroutine
		–	$a_{7:0}$			
RET		48	0100 | 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 | 0011	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 | 1100	RAM(B) → $Q_{3:0}$		
LD	r	–5	00 | r | 0101	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8, A,M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 | y	y → RAM(B) Bd+1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br⊕r → Br	None	Exchange RAM with A, Exclusive-OR Br with r

TABLE III. COP401L-X13/COP401L-R13 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XAD	3,15	23	0010 | 0011	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
		BF	1011 | 1111			
XDS	r	−7	00 | r | 0111	RAM(B) ⟷ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	−4	00 | r | 0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	–	00 | r | (d−1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B immediate with r,d (Note 5)
LEI	y	33	0011 | 0011	y → EN	None	Load EN Immediate (Note 6)
		6–	0110 | y			
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 | 0011		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
		21	0010 | 0001			
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0 = 0$	
	1	11	0001 | 0001		$G_1 = 0$	
	2	03	0000 | 0011		$G_2 = 0$	
	3	13	0001 | 0011		$G_3 = 0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1 = 0$	
	2	03	0000 | 0011		$RAM(B)_2 = 0$	
	3	13	0001 | 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 | 0011	G → A	None	Input G Ports to A
		2A	0010 | 1010			
INL		33	0011 | 0011	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
		2E	0010 | 1110			
OMG		33	0011 | 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 | 1010			
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined) Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1* e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP401L-X13/C0P401L-R13 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM (B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP401L-X13/COP401L-R13 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or will access data in the next group of 4 pages.

COPS Programming Manual

For detailed information on writing COPS programs, the COPS Programming Manual 424410284-001 provides an in-depth discussion of the COPS architecture, instruction set and general techniques of COPS programming. This manual is written with the programmer in mind.

Typical Applications

PROM-Based System

The COP401L-X13/COP401L-R13 may be used to emulate the COP413L. *Figure 9* shows the interconnect to implement a COP401L-X13/COP401L-R13 hardware emulation. This connection uses one MM2716 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP_7–IP_0 are bidirectional inputs and outputs. When the AD/$\overline{DATA}$ clocking output turns on, the EPROM drivers are disabled and IP_7–IP_0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

When AD/$\overline{DATA}$ turns off, the EPROM is enabled and the IP_7–IP_0 pins will input the memory data. P8 outputs the most significant address bit to the memory. (SKIP output may be used for program debug if needed.)

Twenty of the COP401L-X13/COP401L-R13 pins may be configured exactly the same as the COP413L. Selection of the COP401L-X13 or COP401L-R13 depends upon which oscillator is selected for the COP413L.

Oscillator Requirement		Order ROMless
COP413L Option 1 = 0	Ceramic Resonator or external input frequency divided by 8. CKO is oscillator out.	COP401L-X13
Option 1 = 1	Single Pin RC controlled oscillator divided by 4. CKO is no connection.	COP401L-R13

Typical Applications (Continued)

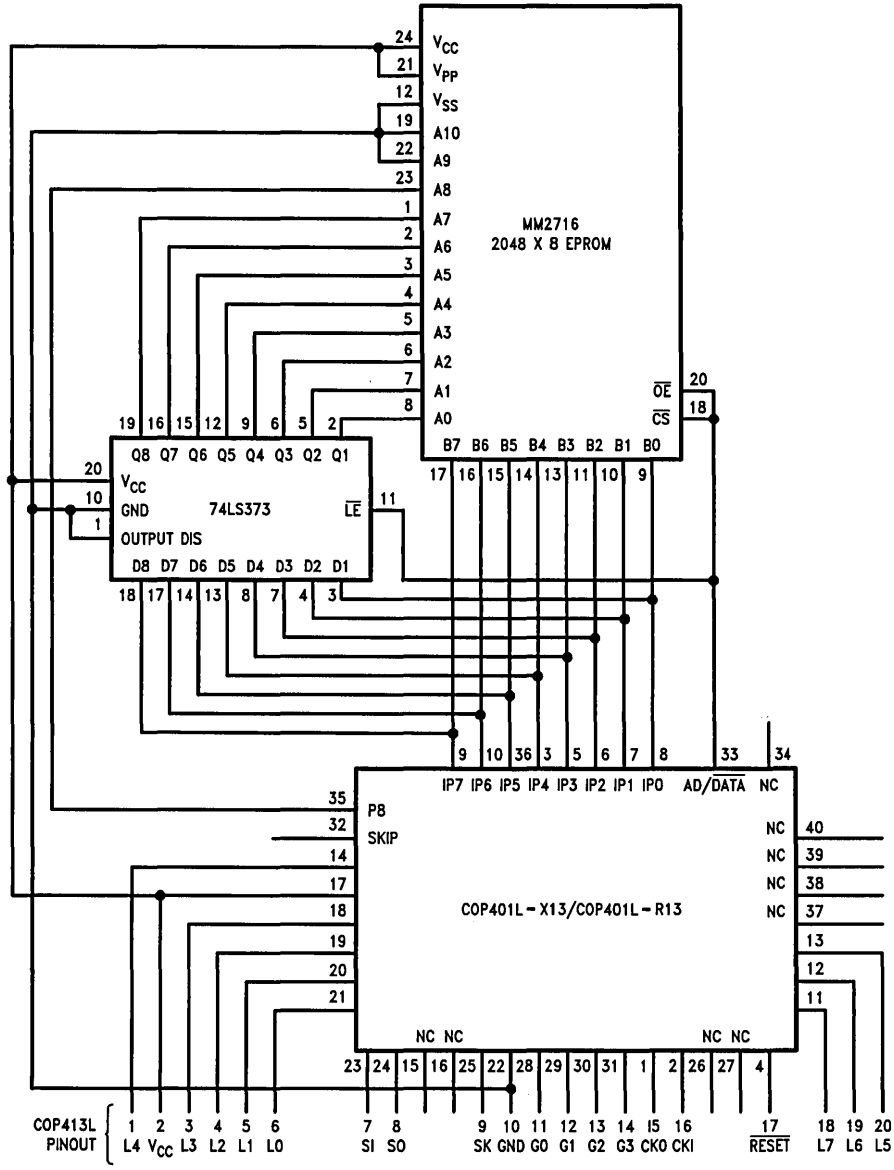


FIGURE 9. COP401L-X13/COP401L-R13 Used to Emulate a COP413L

National Semiconductor

COP402/COP402M ROMless N-Channel Microcontrollers

General Description

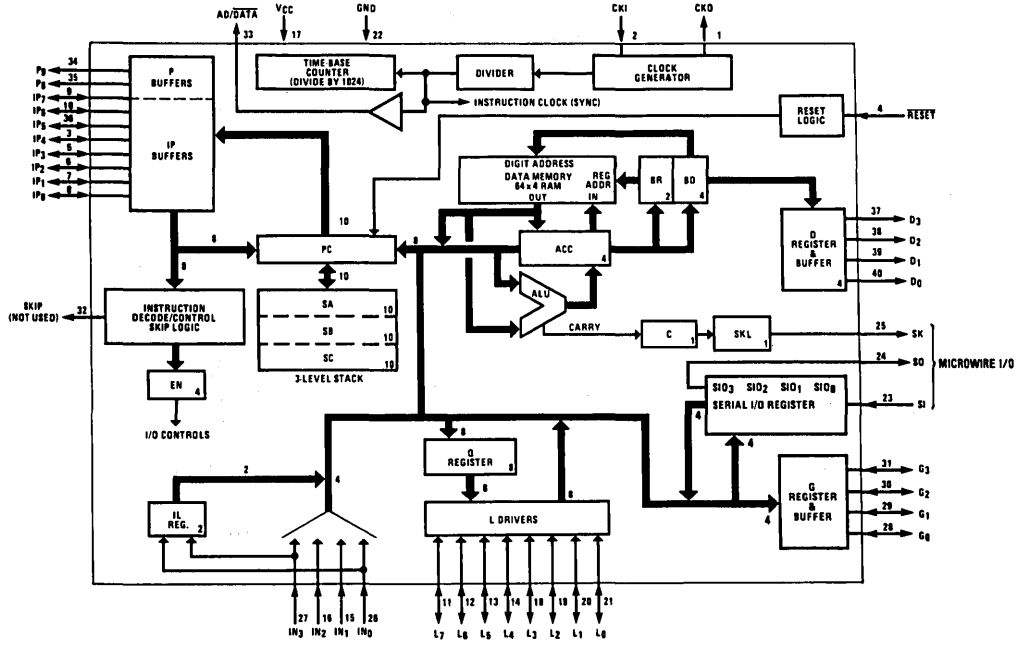
The COP402/COP402M ROMless Microcontrollers are members of the COPS™ family, fabricated using N-channel silicon gate MOS technology. Each part contains CPU, RAM, and I/O, and is identical to a COP420 device, except the ROM has been removed; pins have been added to output the ROM address and to input ROM data. In a system, the COP402 or 402M will perform exactly as the COP420; this important benefit facilitates development and debug of a COP420; this important benefit facilitates development and debug of a COP420 program prior to masking the final part. These devices are also appropriate in low volume applications, or when the program may require changing. The COP402M is identical to the COP402, except the MICROBUS™ interface option has been implemented.

The COP402 may also be used to emulate the COP410L, 411L, or 420L by appropriately reducing the clock frequency.

Features

- Extended temperature (−40°C to +85°C) COP302/COP302M, available as special order
- Low cost
- Exact circuit equivalent of COP420
- Standard 40-pin dual-in-line package
- Interfaces with standard PROM or ROM
- 64 x 4 RAM, addresses up to 1k x 8 ROM
- MICROBUS compatible (COP402M)
- Powerful instruction set
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 4.0 μs instruction time
- Single supply operation (4.5V to 6.3V)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ serial I/O capability
- Software/hardware compatible with other members of COP400 family

Block Diagram



TL/DD/6915-1

FIGURE 1

COP402/COP402M and COP302/COP302M

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin	−0.3V to +7V
Operating Temperature Range	
COP402/COP402M	0°C to 70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 sec.)	300°C
Package Power Dissipation	750 mW at 25°C 400 mW at 70°C 250 mW at 85°C
Total Sink Current	50 mA
Total Source Current	70 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

COP402/COP402M
DC Electrical Characteristics $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	6.3	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	All Outputs Open V_{CC} = 5V		40	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High		2.4		V
Logic Low		−0.3	0.4	V
Schmitt Trigger Input				
RESET				
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Load Source Current	V_{CC} = 5V, V_{IN} = 0V	−100	−800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage	V_{CC} = 5V	−1	+1	μA
Output Voltage Levels				
D, G, L, SK, SO Outputs				
TTL Operation	V_{CC} = 5V ±10%			
Logic High	I_{OH} = −100 μA	2.4		V
Logic Low	I_{OL} = 1.6 mA	−0.3	0.4	V
IP0–IP7, P8, P9, SKIP, CKO, AD/$\overline{DATA}$				
Logic High	I_{OH} = −75 μA	2.4		V
Logic Low	I_{OL} = 400 μA	−0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = 10 μA	−0.3	0.2	V
Output Current Levels				
LED Direct Drive (COP402)	V_{CC} = 6V			
Logic High	V_{OH} = 2.0V	2.5	14	mA
TRI-STATE® (COP402M) Leakage Current	V_{CC} = 5V	−50	+50	μA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

Note 1: TRI-STATE and LED configurations are excluded.

COP402/COP402M
AC Electrical Characteristics $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	÷16 Mode	1.6	4.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time	Frequency = 4 MHz		60	ns
Fall Time	Frequency = 4 MHz		40	ns
Inputs:				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
Output Propagation Delay	Test Conditions: $R_L = 5k$, $C_L = 50$ pF, $V_{OUT} = 1.5V$			
SO and SK				
t_{pd1}			1.0	μs
t_{pd0}			1.0	μs
CKO				
t_{pd1}			0.25	μs
t_{pd0}			0.25	μs
AD/$\overline{DATA}$, SKIP				
t_{pd1}			0.6	μs
t_{pd0}			0.6	μs
All Other Outputs				
t_{pd1}			1.4	μs
t_{pd0}			1.4	μs
MICROBUS Timing	$C_L = 100$ pF, $V_{CC} = 5V \pm 5\%$			
Read Operation *(Figure 4)*				
Chip Select Stable before $\overline{RD}$—t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$—t_{RCS}		20		ns
$\overline{RD}$ Pulse Width—t_{RR}		400		ns
Data Delay from $\overline{RD}$—t_{RD}			375	ns
$\overline{RD}$ to Data Floating—t_{DF}			250	ns
Write Operation *(Figure 5)*				
Chip Select Stable before $\overline{WR}$—t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$—t_{WCS}		20		ns
$\overline{WR}$ Pulse Width—t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$—t_{DW}		320		ns
Data Hold Time for $\overline{WR}$—t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$—t_{WI}			700	ns

Note 1: Duty Cycle = $t_{WI}/(t_{W1} + t_{W0})$.

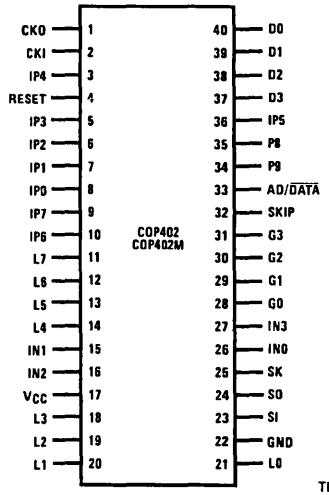
Note 2: See *Figure 9* for additional I/O characteristics.

Note 3: Voltage change must be less than 0.5V in a 1 ms period.

Note 4: Exercise great care not to exceed maximum device power dissipation limits when direct driving LEDs (or sourcing similar loads) at high temperature.

Connection Diagram

Dual-In-Line Package



TL/DD/6915-2

Top View

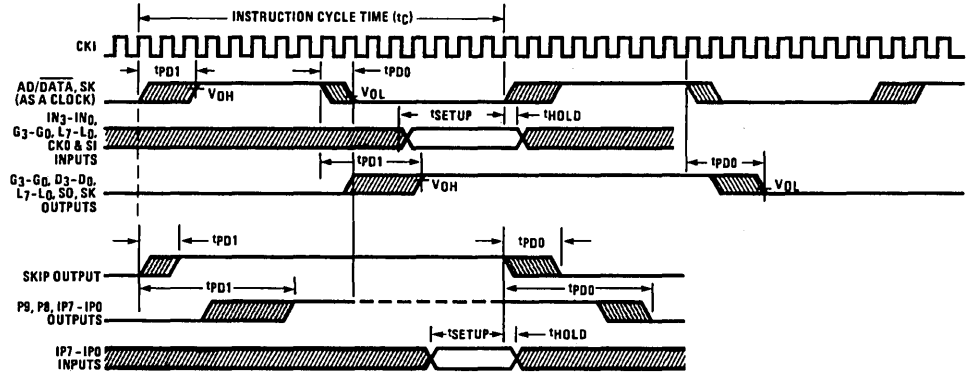
Order Number COP402N or COP402MN
See NS Package Number N40A

FIGURE 2.

Pin Descriptions

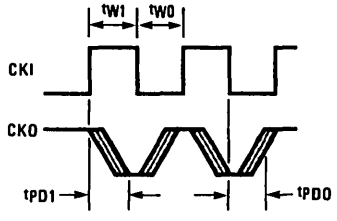
Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{DATA}$	Address out/data in flag
SKIP	Instruction skip output
CKI	System oscillator input
CKO	System oscillator output
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8, P9	2 most significant ROM address outputs

Timing Diagrams



TL/DD/6915-3

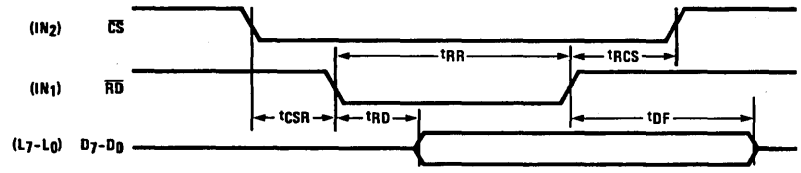
FIGURE 3a. Input/Output Timing Diagrams (Crystal ÷ 16 Mode)



TL/DD/6915-4

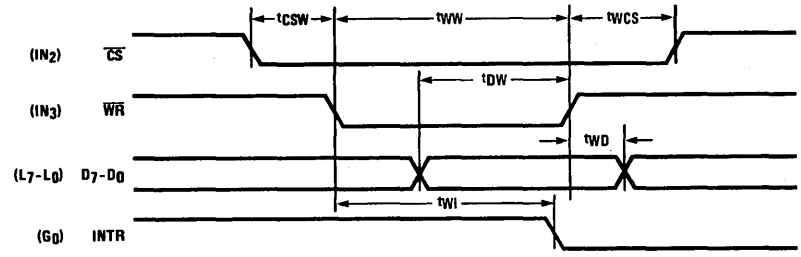
FIGURE 3b. CKO Output Timing

Timing Diagrams (Continued)



TL/DD/6915-5

FIGURE 4. MICROBUS Read Operation Timing



TL/DD/6915-6

FIGURE 5. MICROBUS Write Operation Timing

Functional Description

A block diagram of the COP402 is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024-byte external memory (typically PROM). Words of this memory may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binay value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential **10-bit binary count** value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instruction based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit **A register** (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A **4-bit adder** performs the arithmetic and logic functions of the COP402/402M, storing its results in A. It also outputs a carry bit to the 1-bit **C register**, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four **general-purpose inputs, IN_3–IN_0**, are provided; IN_1, IN_2, and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select and Write Strobe inputs, respectively, for use in MICROBUS applications.

The **D register** provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The **G register** contents are outputs to 4 general-purpose bidirectional I/O ports. G_0 may be mask-programmed as a "ready" output for MICROBUS applications.

The **Q register** is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.) With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

Functional Description (Continued)

The **8 L drivers**, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The **SIO register** functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The **XAS Instruction** copies C into the SKL latch. In the counter mode, SK is the output of SKL. In the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The **EN register** is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state. If the MICROBUS option is being used, EN_2 does not affect the L drivers.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0.

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:
   1. EN_1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon the popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. At *this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and the LQID instruction should not be nested within the interrupt servicing routine since their popping of the stack enables any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. An LEI instruction can be put immediately before the RET to re-enable interrupts.

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

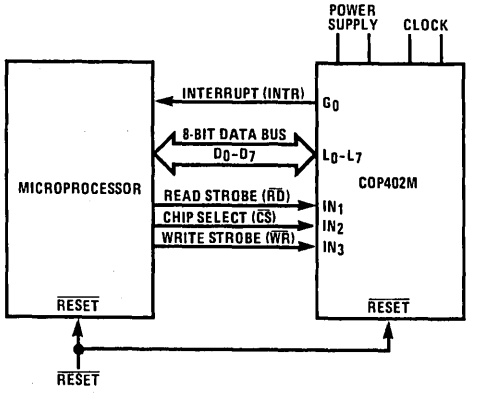
EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = SYNC If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = SYNC If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

MICROBUS INTERFACE

The COP402M can be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN_1, IN_2, and IN_3 general purpose inputs become MICROBUS compatible read-strobe, chip-select, and write-strobe lines, respectively. IN_1 becomes $\overline{RD}$—a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN_2 becomes $\overline{CS}$—a logic "0" on this line selects the COP402M as the μP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN_3 becomes $\overline{WR}$—a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP402M. G_0 becomes INTR, a "ready" output reset by a write pulse from the μP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP402M.

This option has been designed for compatibility with National's MICROBUS—a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS, National Publication.) The functioning and timing relationships between the COP402M signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection to the MICROBUS is shown in *Figure 6*.



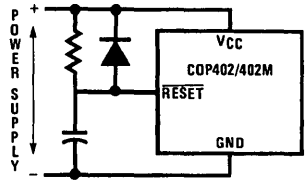
TL/DD/6915-7

FIGURE 6. MICROBUS Option Interconnect

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{RESET}$ pin as shown below. The $\overline{RESET}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{RESET}$ input, provided it stays low for at least two instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, G, and SO are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



$RC \geq 5 \times$ Power Supply Rise Time TL/DD/6915-8

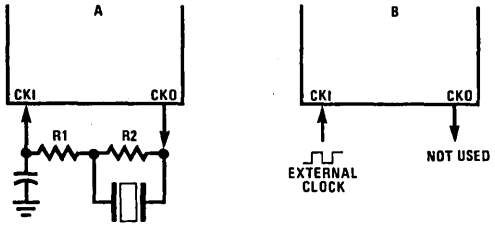
FIGURE 7. Power-Up Clear Circuit

OSCILLATOR

There are two basic clock oscillator configurations available as shown by *Figure 8*.

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16.

b. External Oscillator. CKI is driven by an external clock signal. The instruction cycle time is the clock frequency divided by 16.



TL/DD/6915-9

Crystal Value	Component Values		
	R1	R2	C
4 MHz	1k	1M	27 pF
3.58 MHz	1k	1M	27 pF
2.09 MHz	1k	1M	56 pF

FIGURE 8. COP402/402M Oscillator

EXTERNAL MEMORY INTERFACE

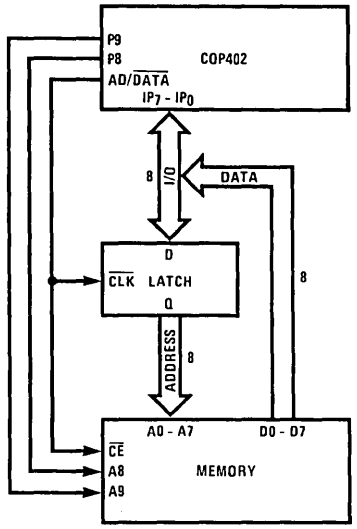
The COP402 and COP402M are designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL = compatible inputs
4. access time = 1.0 μs, max.

Typically these requirements are met using bipolar or MOS PROMs.

Functional Description (Continued)

During operation, the address of the next instruction is sent out on P9, P8, and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P9 and P8 are dedicated address outputs, and do not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input. A simplified block diagram of the external memory interface is shown in *Figure 9*.



TL/DD/6915-10

FIGURE 9. External Memory Interface to COP402

INPUT/OUTPUT

COP402 outputs have the following configurations, illustrated in *Figure 10*.

a. Standard—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements.

b. High Drive—same as **a.** except greater current sourcing capability.

c. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

d. LED Direct Drive—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display.

e. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC} intended to meet the requirements associated with the MICROBUS option. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers.

f. Inputs have an on-chip depletion load device to V_{CC}, as shown in *Figure 10f*.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 10* for each of these devices.

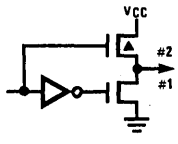
The SO, SK outputs are configured as shown in *Figure 10c*. The D and G outputs are configured as shown in *Figure 10a*.

Functional Description (Continued)

Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs are configured as in *Figure 10d* on the COP402. On the COP402M the L outputs are as in *Figure 10e*.

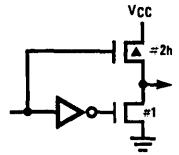
An important point to remember if using configuration d with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 11.*)

IP7 through IP0 outputs are configured as shown in *Figure 10c*; P9, P8, SKIP, and AD/$\overline{\text{DATA}}$ are configured as shown in *Figure 10b.*



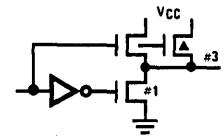
TL/DD/6915–11

a. Standard



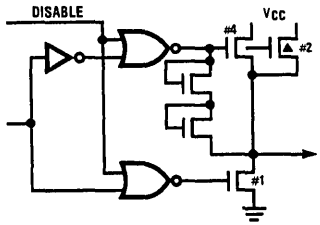
TL/DD/6915–12

b. High Drive



TL/DD/6915–13

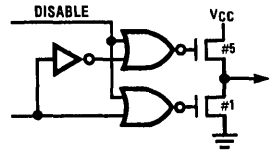
c. Push-Pull



TL/DD/6915–14

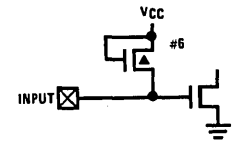
d. LED

(▲ is Depletion Device)



TL/DD/6915–15

e. TRI-STATE Push-Pull



TL/DD/6915–16

f. Input with Load

FIGURE 10. Input/Output Configurations

Typical Performance Characteristics

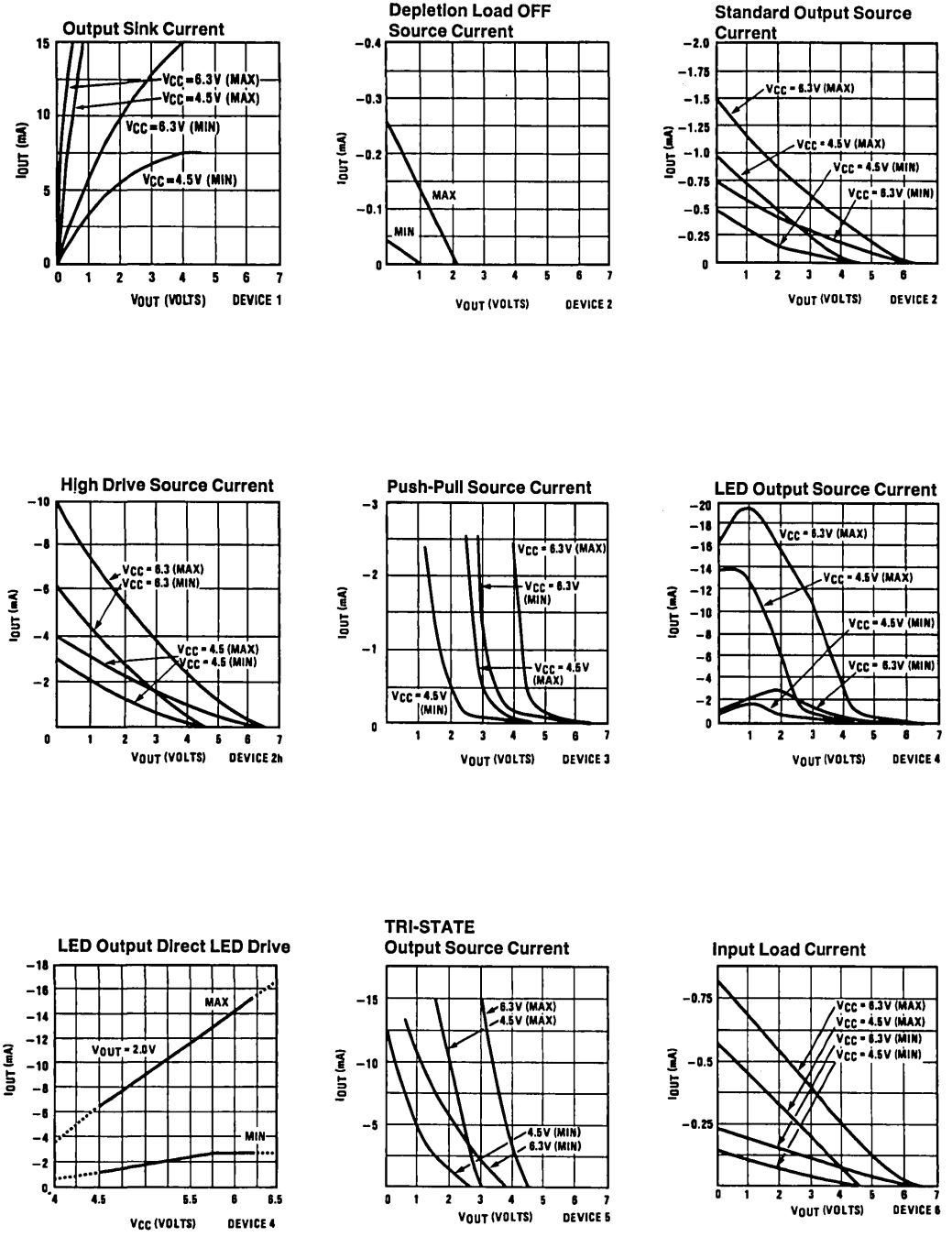
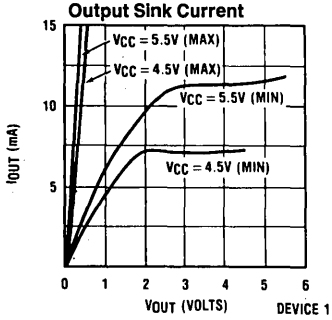


FIGURE 11. COP402/COP402M Input/Output Characteristics

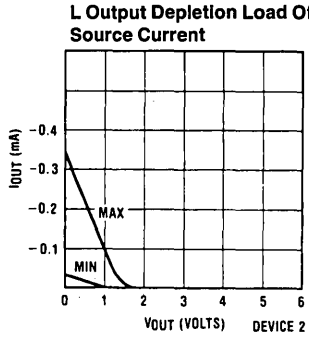
TL/DD/6915–17

Typical Performance Characteristics (Continued)

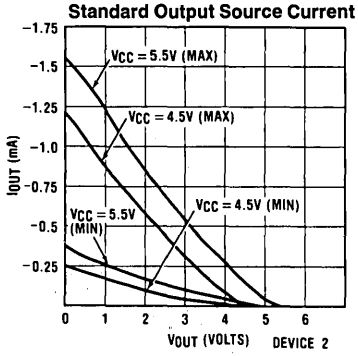
Output Sink Current



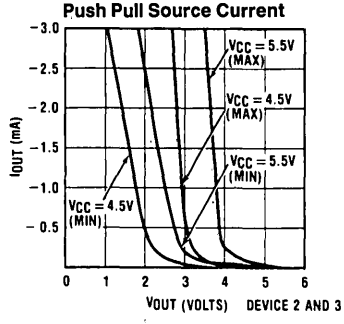
L Output Depletion Load Off Source Current



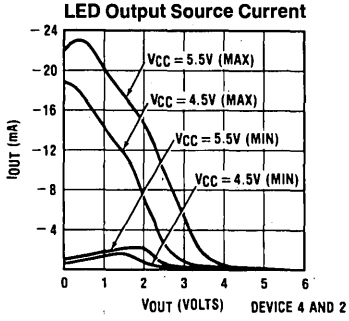
Standard Output Source Current



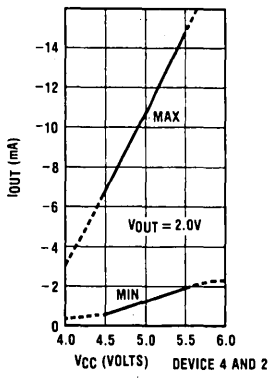
Push Pull Source Current



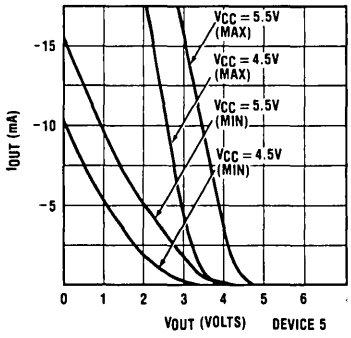
LED Output Source Current



LED Output Device LED Drive



TRI-STATE Output Source Current



Input Load Source Current

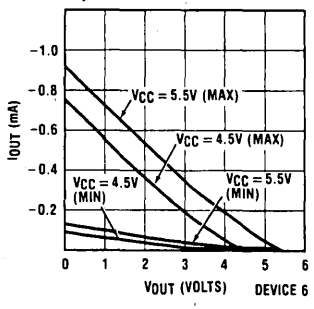


FIGURE 11a. COP302/COP302M Input/Output Characteristics

TL/DD/6915–18

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP402/402M instruction set.

TABLE II. COP402/COP402M Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit Latches Associated with the IN_3 or IN_0 inputs
IN	4-bit Input port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
P	2-bit ROM Address Port
PC	10-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP402/COP402M Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP402/COP402M Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 | 1111	ROM ($PC_{9:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6–	0110 | 00 | $a_{9:8}$	a → PC	None	Jump
		--	$a_{7:0}$			
JP	a	--	1 | $a_{6:0}$ (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 4)
		--	11 | $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	--	10 | $a_{5:0}$	PC + 1 → SA → SB → SC 0010 → $PC_{9:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6–	0110 | 10 | $a_{9:8}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
		--	$a_{7:0}$			
RET		48	0100 | 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 | 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 | 0011	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
		3C	0011 | 1100			
CQMA		33	0011 | 0011	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
		2C	0010 | 1100			
LD	r	–5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23	0010 | 0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		--	00 | r | d			
LQID		BF	1011 | 1111	ROM($PC_{9:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 | 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → $RAM(B)_1$		
	2	42	0100 | 0010	0 → $RAM(B)_2$		
	3	43	0100 | 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 | 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → $RAM(B)_1$		
	2	46	0100 | 0110	1 → $RAM(B)_2$		
	3	4B	0100 | 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 | r | 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	0010 | 0011	RAM(r,d) ⟷ A	None	Exchange A with RAM pointed to directly by r,d
		--	10 | r | d			

Instruction Set (Continued)

TABLE III. COP402/COP402M Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	–7	|00|r|0111|	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	|00|r|0100|	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	– –	|00|r|(d – 1)| (d=0, 9:15) or |0011|0011| |10|r|d| (any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6–	|0011|0011| |0110|y|	y → EN	None	Load EN Immediate (Note 7)
XABR		12	|0001|0010|	A ⟷ Br (0,0 → A_3,A_2)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	G_0 = 0	
	1	11	|0001|0001|		G_1 = 0	
	2	03	|0000|0011|		G_2 = 0	
	3	13	|0001|0011|		G_3 = 0	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1$ = 0	
	2	03	|0000|0011|		$RAM(B)_2$ = 0	
	3	13	|0001|0011|		$RAM(B)_3$ = 0	
SKT		41	|0100|0001|		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP402/COP402M Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 | 0011 0010 | 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 | 0011 0010 | 1000	IN → A	None	Input IN Inputs to A (Notes 2 and 8)
INIL		33 29	0011 | 0011 0010 | 1001	IL_3, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 | 0011 0010 | 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM,A
OBD		33 3E	0011 | 0011 0011 | 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5–	0011 | 0011 0101 | y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit register.

Note 2: The ININ instruction is not available on the 24-pin COP421 since this device does not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: The COP402M will always read a "1" into A1 with the ININ instruction.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register.

The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once evey 4 instruction cycles to effect a continuous data stream.

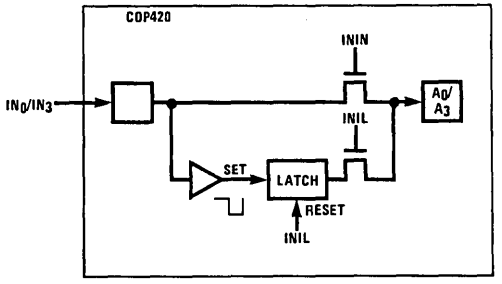
JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, $PC_{9:8}$, A, M. PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 12*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IN_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon the execution of an ININ instruction. (See Table III, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.



TL/DD/6915–19

FIGURE 12. IN_0/IN_3 Latches

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip on Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the controller to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 131 kHz (crystal frequency ÷ 16) and the binary counter output pulse frequency will be 128 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 128 ticks.

INSTRUCTION SET NOTES

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed, except JID and LQID. LQID and JID take two cycle times if executed and one if skipped.

c. The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, or 15 will access data in the next group of 4 pages.

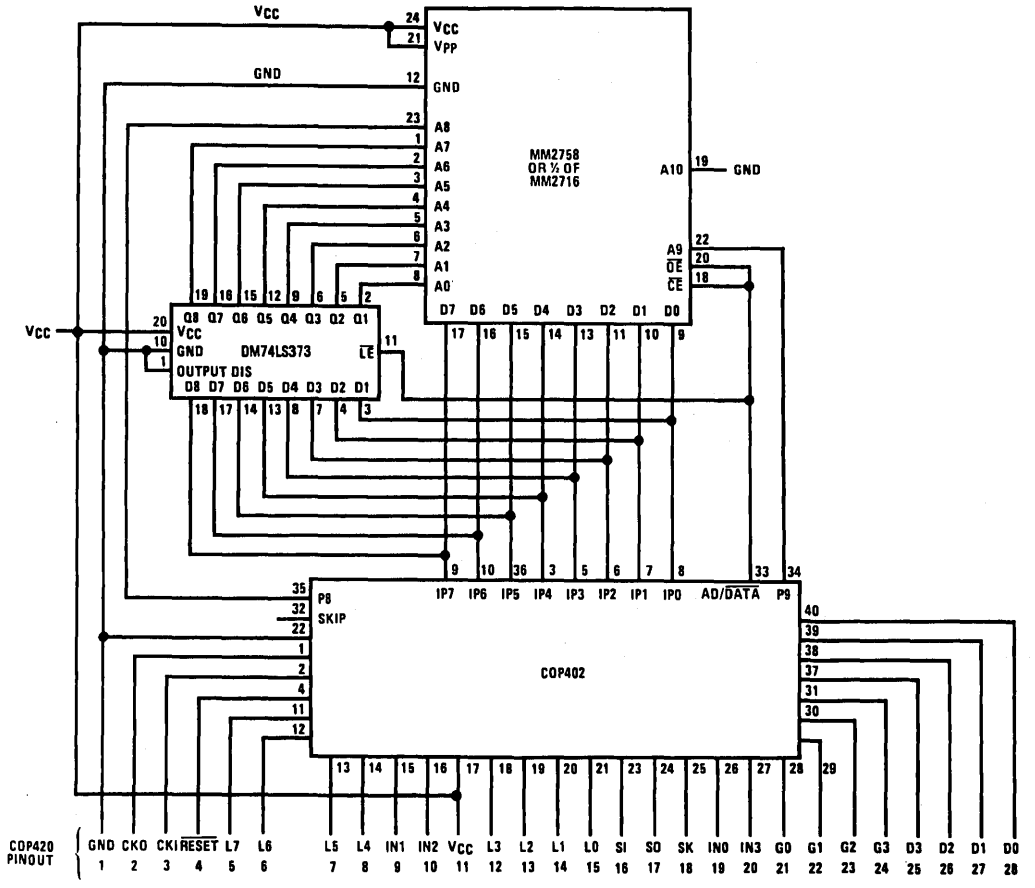
Typical Application: PROM-Based System

The COP402 may be used to exactly emulate the COP420, *Figure 13* shows the interconnect to implement a COP420 hardware emulation. This connection uses two MM5204 EPROMs as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

When AD/$\overline{\text{DATA}}$ turns off, the EPROMs are enabled and the IP7-IP0 pins will input the memory data. P8 and P9 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

The other 28 pins of the COP402 may be configured exactly the same as a COP420. The COP402M chip can be used if the MICROBUS feature of the COP420 is needed.



TL/DD/6915–20

FIGURE 13. COP402 Used to Emulate a COP420

Option List

COP402 MASK OPTIONS

The following COP420 options have been implemented in this basic version of the COP402. Subsequent versions of the COP402 will implement different combinations of available options; such versions will be identified as COP402-A, COP402-B, etc.

Option Value	Comment
Option 1 = 0	Ground Pin—no option available
Option 2 = 0	CKO is clock generator output to crystal
Option 3 = 0	CKI is crystal input ÷16 (may be overridden externally)
Option 4 = 0	RESET pin has load device to V_{CC}
Option 5 = 2 (402) = 3 (402M)	L7 has LED direct-drive output L7 has TRI-STATE push-pull output
Option 6 = 2, 3	L6 same as L7
Option 7 = 2, 3	L5 same as L7
Option 8 = 2, 3	L4 same as L7
Option 9 = 0 (402) = 1 (402M)	IN1 has load device to V_{CC} Hi Z
Option 10 = 0 (402) = 1 (402M)	IN2 has load device to V_{CC} Hi Z
Option 11 = 0	V_{CC} pin—no option available
Option 12 = 2, 3	L3 same as L7
Option 13 = 2, 3	L2 same as L7
Option 14 = 2, 3	L1 same as L7
Option 15 = 2, 3	L0 same as L7
Option 16 = 0	SI has load device to V_{CC}
Option 17 = 2	SO has push-pull output
Option 18 = 2	SK has push-pull output
Option 19 = 0	IN0 has load device to V_{CC}
Option 20 = 0 (402) = 1 (402M)	IN3 has load device to V_{CC} Hi Z
Option 21 = 0	G0 has standard output
Option 22 = 0	G1 same as G0
Option 23 = 0	G2 same as G0
Option 24 = 0	G3 same as G0
Option 25 = 0	D3 has standard output
Option 26 = 0	D2 same as D3
Option 27 = 0	D1 same as D3
Option 28 = 0	D0 same as D3
Option 29 = 0 (402) = 1 (402M)	normal operation MICROBUS operation
Option 30 = N/A	40-pin package

National Semiconductor

COP404 ROMless N-Channel Microcontroller

General Description

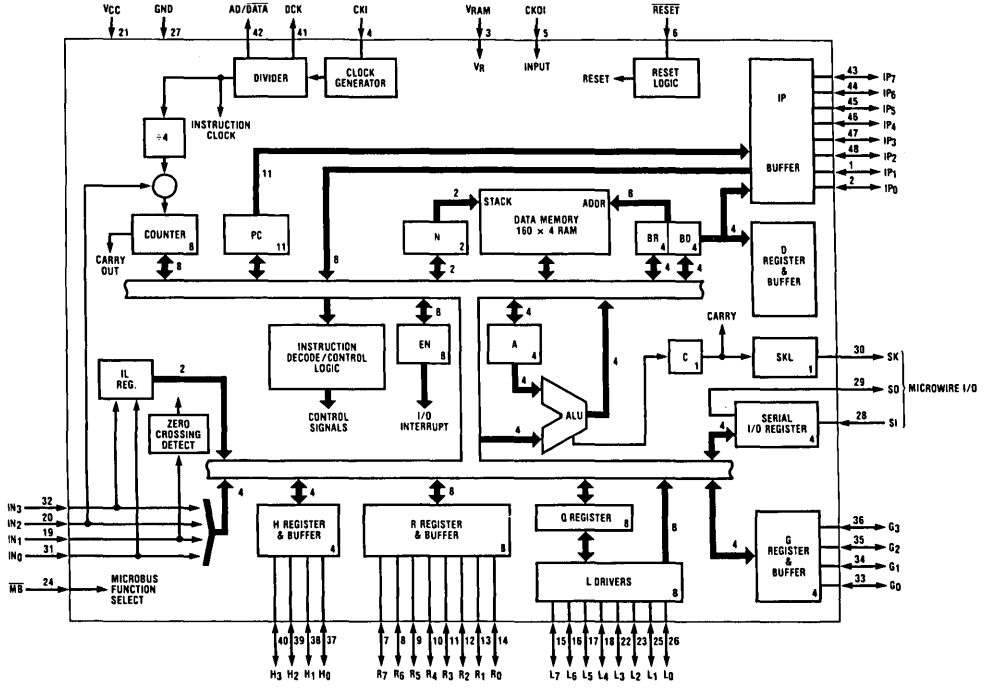
The COP404 ROMless N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. Each microcontroller contains all system timing, internal logic, RAM and I/O necessary to implement dedicated control functions in a variety of applications, and is identical to the COP440/COP340 devices, except that the ROM has been removed; pins have been added to output the ROM address and to input ROM data. In a system, the COP404 will perform exactly as the COP440; this important benefit facilitates development and debug of a COP440 program prior to masking the final part. Features include single supply operation, various output configurations, and an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output and data manipulation. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a controller-oriented processor at a low end-product cost.

For extended temperature range (−40°C to +85°C) COP304 available on special order.

Features

- Exact circuit equivalent of COP440
- Standard 48-pin dual-in-line package
- Interfaces with standard PROM or ROM
- Enhanced, more powerful instruction set
- 160 × 4 RAM, addresses up to 2k × 8 ROM
- MICROBUS™ compatible
- Zero-crossing detect circuitry with hysteresis
- True multi-vectored interrupt from four selectable sources (plus restart)
- Four-level subroutine stack (in RAM)
- 4 μs cycle time
- Single supply operation (4.5V–6.3V)
- Programmable time-base counter for real-time processing
- Internal binary counter/register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- Software/hardware compatible with other members of COP400 family
- Compatible dual CPU device available

Block Diagram



TL/DD/6916–1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Zero-Crossing Detect Pin Relative to GND	−1.2V to +15V
Voltage at Any Other Pin Relative to GND	−0.5V to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	150 mA
Total Sink Current	90 mA

Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 4)	4.5	6.3	V
Power Supply Ripple	(Peak to Peak)		0.4	V
Operating Supply Current	(All Inputs and Outputs Open)			
	$T_A = 0°C$		44	mA
	$T_A = 25°C$		37	mA
	$T_A = 70°C$		30	mA
V_R RAM Power Supply Current	$V_R = 3.3V$		3	mA
Input Voltage Levels				
CKI Input Levels (÷16)				
Logic High (V_{IH})	V_{CC} = Max.,	2.5		V
Logic High (V_{IH})	$V_{CC} = 5V \pm 5\%$	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
Zero-Crossing Detect Input (IN_1)	Zero-Crossing Interrupt Input; INIL Instruction			
Trip Point		−0.15	0.15	V
Logic High (V_{IH}) Limit			12	V
Logic Low (V_{IL}) Limit		−0.8		V
IN_1				
Logic High	Interrupt Input;			
	ININ Instruction;	3.0		V
Logic Low	MICROBUS Input	−0.3	0.8	V
All Other Inputs				
Logic High	V_{CC} = Max.	2.5		V
Logic High	$V_{CC} = 5V \pm 5\%$	2.0		V
Logic Low		−0.3	0.8	V
IN_1 Input Resistance to Ground	$V_{IH} = 1.0V$	1.5	4.6	kΩ
Input Load Source Current	$V_{IH} = 2.0V$, $V_{CC} = 4.5V$	14	230	μA
Input Capacitance			7.0	pF
Hi-Z Input Leakage		−1.0	+1.0	μA
Output Voltage Levels				
Standard Output				
TTL Operation				
Logic High (V_{OH})	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low (V_{OL})	$I_{OL} = 1.6$ mA		0.4	V
CMOS Operation (Note 1)				
Logic High (V_{OH})	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.4$		V
Logic Low (V_{OL})	$I_{OL} = 10\ \mu A$		0.2	V
TRI-STATE Output				
TTL Operation				
Logic High (V_{OH})	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low (V_{OL})	$I_{OL} = 1.6$ mA		0.4	V
CMOS Operation (Note 1)	$33\ k\Omega \ge R_L \ge 4.7\ k\Omega$			
Logic High (V_{OH})	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.5$		V
Logic Low (V_{OL})	$I_{OL} = 1.6$ mA		0.4	V
Output Current Levels				
Standard Output Source Current	$V_{CC} = 4.5V$, $V_{OH} = 2.4V$	−100	−650	μA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Total Sink Current Allowed				
All I/O Combined			90	mA
Each L, R Port			20	mA
Each D, G, H Port			10	mA
SO, SK			2.5	mA
IP			1.8	mA
Total Source Current Allowed	(Note 5)			
All I/O Combined			150	mA
L Port			120	mA
L_7–L_4			70	mA
L_3–L_0			70	mA
Each L Pin			23	mA
All Other Output Pins			1.6	mA

Note 1: TRI-STATE configuration is excluded.

AC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time—t_E		4.0	10	μs
CKI Frequency	÷16 Mode	1.6	4.0	MHz
Duty Cycle (Note 2)	f_I = 4 MHz	30	60	%
Rise Time	f_I = 4 MHz		60	ns
Fall Time	f_I = 4 MHz		40	ns
INPUTS: *(Figure 3)*				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		300		ns
IP				
t_{SETUP}		0.25		μs
t_{HOLD}		250		ns
t_{HOLD}	From AD/$\overline{DATA}$ Rising Edge	0		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
OUTPUT PROPAGATION DELAY	Test Condition:			
IP	C_L = 50 pF, V_{OUT} = 1.5V			
t_{pd1A}, t_{pd0A}			1.94	μs
t_{pd1B}, t_{pd0B}			0.94	μs
DCK				
t_{pd1}, t_{pd0}			375	ns
AD/$\overline{DATA}$				
t_{pd1}, t_{pd0}			300	ns
SO, SK				
t_{pd1}, t_{pd0}	R_L = 2.4 kΩ		1.0	μs
All Other Outputs	R_L = 5.0 kΩ		1.4	μs
MICROBUS TIMING	C_L = 100 pF, V_{CC} = 5V ± 5%			
Read Operation	TRI-STATE outputs			
Chip Select Stable Before $\overline{RD}$—t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$—t_{RCS}		20		ns
$\overline{RD}$ Pulse Width—t_{RR}		400		ns
Data Delay from $\overline{RD}$—t_{RD}			375	ns
$\overline{RD}$ to Data Floating—t_{DF}			250	ns
Write Operation				
Chip Select Stable Before $\overline{WR}$—t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$—t_{WCS}		20		ns
$\overline{WR}$ Pulse Width—t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$—t_{DW}		320		ns
Data Hold Time for $\overline{WR}$—t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$—t_{WI}			700	ns

Note 2: Duty Cycle = $t_{WI}/(t_{WI} + t_{WO})$.

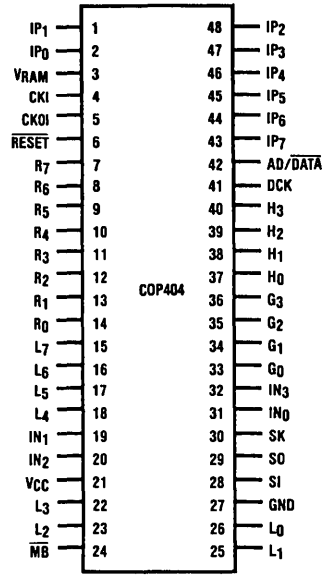
Note 3: See Figure for additional I/O Characteristics.

Note 4: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 5: Exercise great care not to exceed maximum device power dissipation limits when direct-driving LEDs (or sourcing similar loads) at high temperature.

Connection Diagram

Dual-In-Line Package



TL/DD/6916-2

Top View

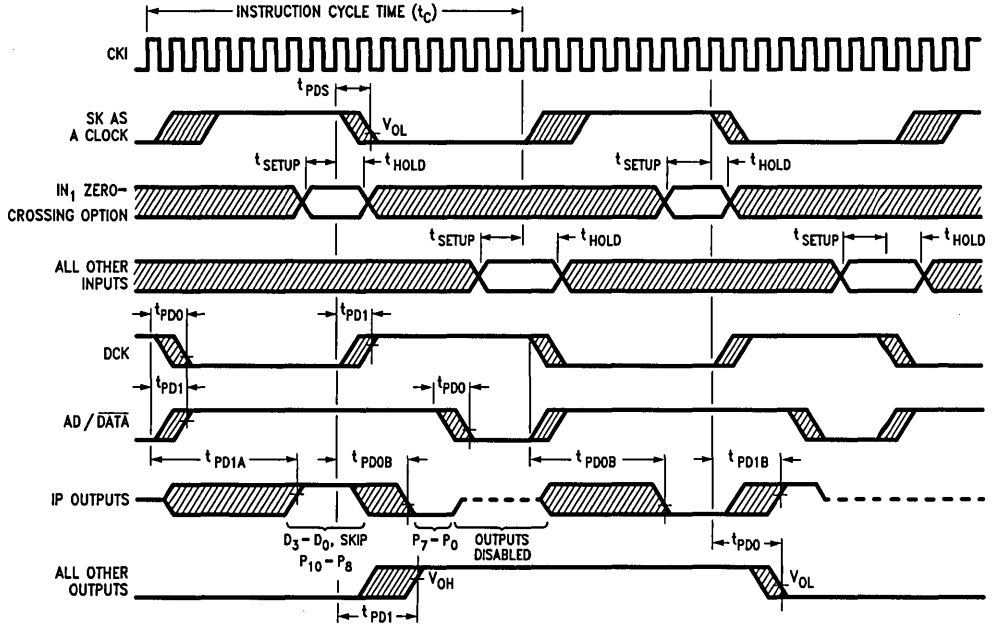
FIGURE 2

Order Number COP404N
See NS Package Number N48A

Pin Descriptions

Pin	Description
L_7–L_0	8-bit bidirectional TRI-STATE I/O port
G_3–G_0	4-bit bidirectional I/O port
IN_3–IN_0	4-bit general purpose input port
H_3–H_0	4-bit bidirectional I/O port.
R_7–R_0	8-bit bidirectional TRI-STATE I/O port
SI	Serial input
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKOI	General purpose input
V_{RAM}	Power supply to first 4 registers of RAM
$\overline{MB}$	MICROBUS function select
DCK	Clock output to latch D outputs and high order address bits
AD/$\overline{DATA}$	Address out/data in flag
IP_1–IP_0	8-bit bidirectional port for ROM address, ROM data and D outputs
$\overline{RESET}$	System reset input
V_{CC}	Power Supply
GND	Ground

Timing Diagram



TL/DD/6916-3

FIGURE 3. Input/Output Timing Diagrams (÷ 16 Mode)

Functional Description

The COP404 is a ROMless microcontroller for emulating the COP440 or for stand-alone applications. Please refer to the COP440 description for detail functional description. The following describes functions that are unique to the COP404 or are different from those in COP440. *Figures 1* and *2* show the COP404 block diagram and pin-out.

PROGRAM MEMORY

Program memory consists of 2048 bytes of external memory (on-chip in the COP440) that can be accessed through the IP port. See External Memory Interface below.

D PORT

The D3–D0 outputs are missing from this 48-pin package, but may be recovered through the IP port (see External Memory Interface below). Note that the recovered signals have the same timing but different output drive capability as those from the COP440 (see D Port Characteristics below).

MICROBUS AND ZERO-CROSSING DETECT INPUT OPTION

The MICROBUS compatible I/O, selected by a mask option on the COP440, is selected by tying the $\overline{MB}$ pin directly to ground. When the MICROBUS compatible I/O is not desired, the $\overline{MB}$ pin should be tied to V_{CC}. Note that none of the IN inputs are Hi-Z. Since zero-crossing detect input (used by INIL instruction and zero-crossing interrupt feature) is chosen for IN1, the IN1 input "1" level for ININ instruction, IN1 interrupt, and MICROBUS input is 3V. Even though the MICROBUS option and zero-crossing detector option appear on the COP404, they are mutually exclusive on the COP440.

OSCILLATOR

CKI is an external clock input signal. The clock frequency is divided by 16 to give the execution frequency.

CKO PIN OPTIONS

Two different CKO functions of the COP440 are available on the COP404. V_{RAM} supplies power to the lower four registers of RAM, and CKOI is an interrupt input or a general purpose input, reading into bit 2 of A (accumulator) through the INIL instruction.

EXTERNAL MEMORY INTERFACE

The COP404 is designed for use with an external program memory. This memory may be implemented using any devices having the following characteristics:

1. Random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL-compatible inputs
4. Access time = 450 ns maximum

Typically these requirements are met using bipolar or MOS PROMs.

Figure 3 shows the timings for IP port and the external memory interface clocks—DCK and AD/$\overline{DATA}$. While DCK is low, the upper three address bits, P10–P8, of the next instruction to be executed appear at IP2–IP0 respectively; D3–D0 appear at IP7–IP4 and IP3 contains the SKIP output used by the COPS Program Development System (PDS). The rising edge of DCK clocks these data into D flip-flops, e.g., 74LS374. The timing of D port data is then the same for COP404 and COP440. After DCK has risen to a "1" level, the remaining address bits (P7–P0) appear at IP7–IP0. The falling edge of AD/$\overline{DATA}$ latches these data into flow-through latches, e.g., 74LS373. The latched addresses provide the inputs to the external memory. When AD/$\overline{DATA}$ goes low, the IP outputs are disabled and the IP lines become program memory inputs from the external memory. Note that DCK has a duty cycle of about 50% and AD/$\overline{DATA}$ has a duty cycle of about 75%. *Figure 4* shows how to emulate the COP440 using a COP404 and an EPROM as the external memory.

I/O OPTIONS

All inputs except IN1 and CKI have on-chip depletion load devices to V_{CC}. IN1 has a resistive load to GND due to the zero-crossing input. CKI is a Hi-Z input.

G and H ports have standard outputs. L and R ports have TRI-STATE outputs. IP port, DCK, AD/$\overline{DATA}$, SO and SK have push-pull outputs.

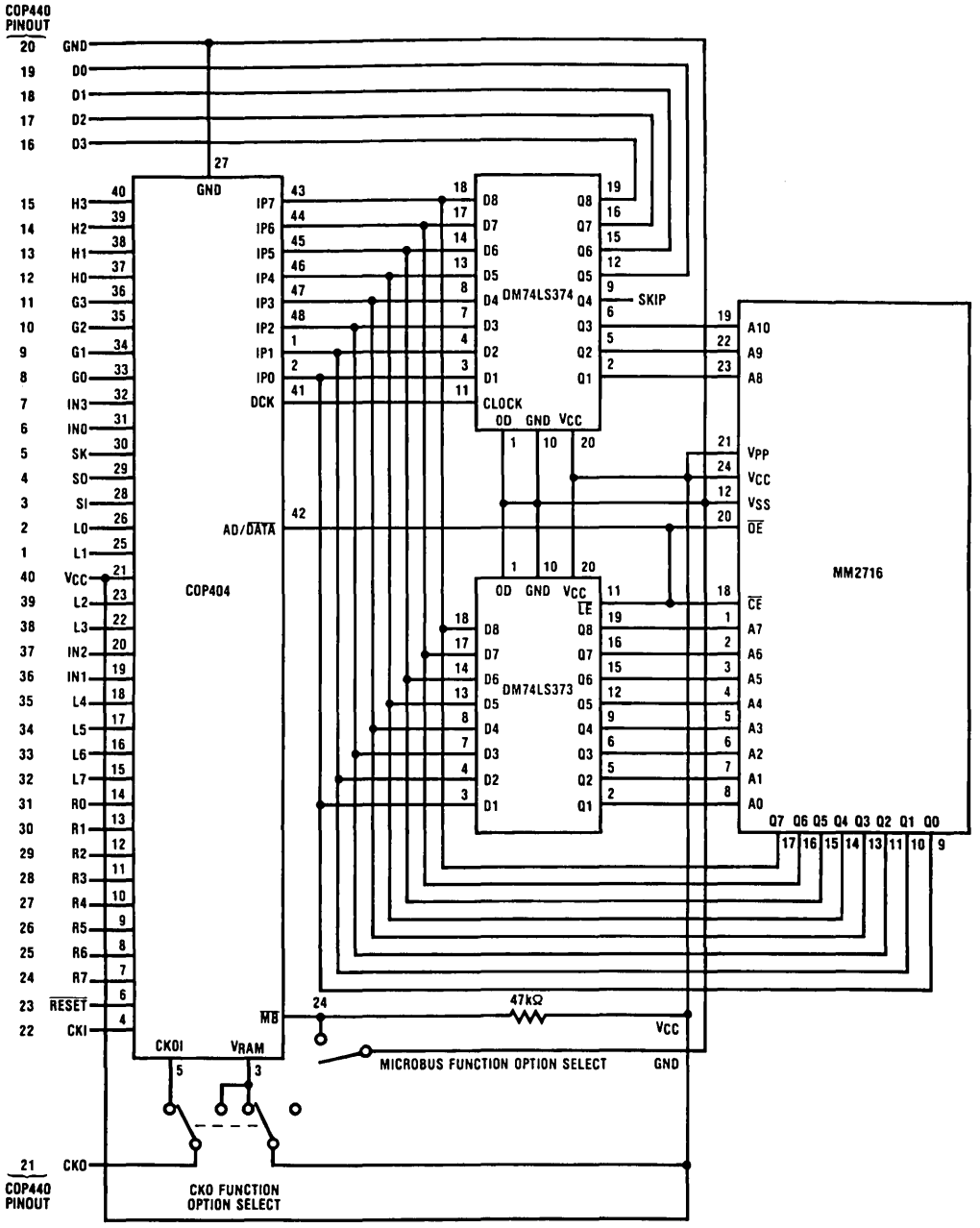
LED DRIVE

The TRI-STATE outputs of L port may be used to drive the segments of an LED display. External current limiting resistors of 100Ω must be connected between the L outputs and the LED segments.

D PORT CHARACTERISTICS

Since the D port is recovered through an external latch, the output drive is that of the latch and not that of COP440. Using the set-up as shown in *Figure 4*, at an output "0" level of 0.4V, the 74LS374 may sink 10 times as much current as the COP440. At an output "1" level of 2.4V, the 74LS374 may source 10 times as much current as the COP440. On the other hand, the output "1" level of 74LS374 latch does not go to V_{CC} without an external pull-up resistor. In order to better approximate the COP440 output characteristics, add a 74C906 buffer to the output of the 74LS374, thus emulating an open drain D output. A pull-up resistor of 10k should be added to the input of the buffer. To emulate the standard output, add a pull-up resistor between 2.7k and 15k to the output of the 74C906.

Functional Description (Continued)



TL/DD/6916–4

FIGURE 4. COP404 Used to Emulate a COP440

Option Table

COP404 MASK OPTIONS

The following COP440 options have been implemented in the COP404.

Option Value		Comment
Option 1–2	= 3	L outputs are TRI-STATE
Option 3	= 0	SI has load to V_{CC}
Option 4	= 2	SO is push-pull output
Option 5	= 2	SK is push-pull output
Option 6	= 0	IN0 has load to V_{CC}
Option 7	= 0	IN3 has load to V_{CC}
Option 8–11	= 0	G outputs are standard
Option 12–15	= 0	H outputs are standard
Option 16–19	= N/A	D outputs are derived from external latch, see *Figure 4*
Option 20	= N/A	GND—No option
Option 21	= 1,2	CKO is replaced by V_{RAM} and CKOI
Option 22	= 0	CKI is input clock divided by 16
Option 23	= 0	$\overline{\text{RESET}}$ has load to V_{CC}
Option 24–31	= 3	R outputs are TRI-STATE
Option 32–35	= 3	L outputs are TRI-STATE
Option 36	= 2	IN1 is zero-crossing detect input
Option 37	= 0	IN2 has load to V_{CC}
Option 38–39	= 3	L outputs are TRI-STATE
Option 40	= N/A	V_{CC}—No option available
Option 41	= 0,1	MICROBUS option is pin selectable
Option 42–48	= 0	Inputs have standard TTL levels
Option 49	= N/A	No option available
Option 50	= N/A	48-pin package

National Semiconductor

COP404C ROMless CMOS Microcontrollers

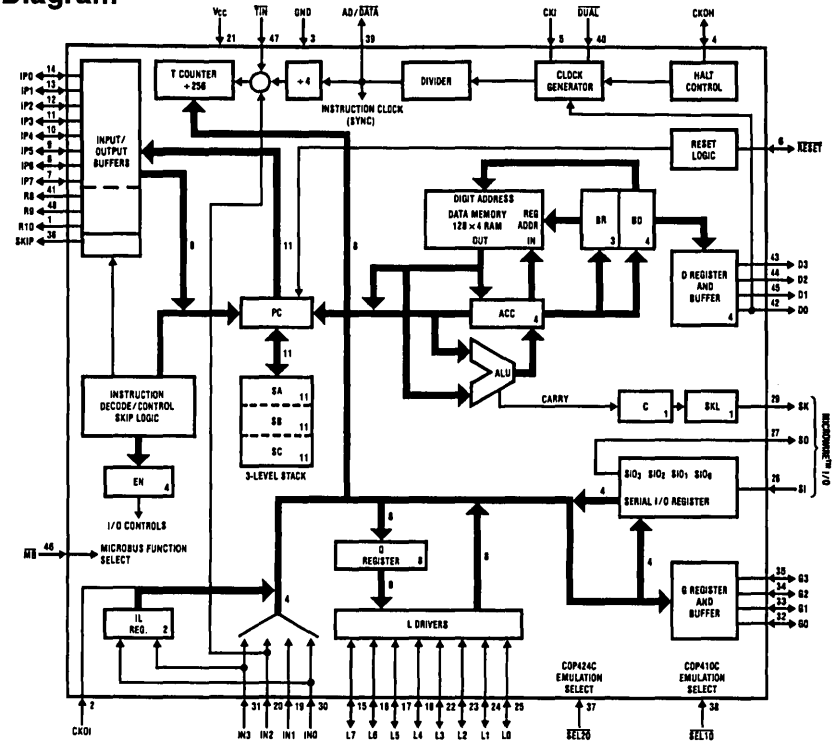
General Description

The COP404C ROMless Microcontroller is a member of the COPS™ family, fabricated using double-poly, silicon gate CMOS (microCMOS) technology. The COP404C contains CPU, RAM, I/O and is identical to a COP444C device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. The COP404C can be configured, by means of external pins, to function as a COP444C, a COP424C, or a COP410C. Pins have been added to allow the user to select the various functional options that are available on the family of mask-programmed CMOS parts. The COP404C is primarily intended for use in the development and debug of a COP program for the COP444C/445C, COP424C/425C, and COP410C/411C devices prior to masking the final part. The COP404C is also appropriate in low volume applications or when the program might be changing.

Features

- Accurate emulation of the COP444C, COP424C and COP410C
- Lowest Power Dissipation (50 μW typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4 μs instruction time, plus software selectable clocks
- 128 × 4 RAM, addresses 2k × 8 ROM
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (2.4V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible
- MICROBUS™ compatible
- Software/hardware compatible with other members of the COP400 family

Block Diagram



TL/DD/5530–1

FIGURE 1. Block Diagram

Absolute Maximum Ratings

Supply Voltage	6V	Operating temperature range	0° to +70°C
Voltage at any pin	−0.3V to V_{CC} + 0.3V	Storage temperature range	−65°C to +150°C
Total Allowable Source Current	25 mA	Lead temperature (soldering, 10 sec.)	300°C
Total Allowable Sink Current	25 mA		

DC Electrical Characteristics $0°C \leq T_a \leq 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple (Note 5)	peak to peak		0.1 V_{CC}	V
Supply Current (Note 1)	V_{CC}=2.4V, t_c=64 μs		120	μA
	V_{CC}=5.0V, t_c=16 μs		700	μA
	V_{CC}=5.0V, t_c=4 μs (T_c is instruction cycle time)		3000	μA
HALT Mode Current (Note 2)	V_{CC}=5.0V, F_{IN}=0 kHz, T_A=25°C		20	μA
	V_{CC}=2.4V, F_{IN}=0 kHz, T_A=25°C		6	μA
Input Voltage Levels				
RESET, D0 (clock input)				
CKI				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All other inputs (Note 7)				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-up current	V_{CC}=4.5V, V_{IN}=0	30	330	μA
Hi-Z input leakage		−1	+1	μA
Input capacitance (Note 4)			7	pF
Output Voltage Levels	Standard outputs			
LSTTL Operation	V_{CC}=5.0V ±10%			
Logic High	I_{OH}=−100 μA	2.7		V
Logic Low	I_{OL}=400 μA		0.4	V
CMOS Operation				
Logic High	I_{OH}=−10 μA	V_{CC}−0.2		V
Logic Low	I_{OL}=10 μA		0.2	V
Output current levels				
Sink (Note 6)	V_{CC}=4.5V, V_{OUT}=V_{CC}	1.2		mA
	V_{CC}=2.4V, V_{OUT}=V_{CC}	0.2		mA
Source (Standard option)	V_{CC}=4.5V, V_{OUT}=0V	0.5		mA
	V_{CC}=2.4V, V_{OUT}=0V	0.1		mA
Source (Low current option)	V_{CC}=4.5V, V_{OUT}=0V	30	330	μA
	V_{CC}=2.4V, V_{OUT}=0V	6	80	μA
Allowable Sink/Source current per pin (Note 6)			5	mA
Allowable Loading on CKOH			100	pF
Current needed to over-ride HALT (Note 3)				
To continue	V_{CC}=4.5V, V_{IN}=2V_{CC}		.7	mA
To halt	V_{CC}=4.5V, V_{IN}=7V_{CC}		1.6	mA
TRI-STATE leakage current		−2.5	+2.5	μA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP404C

AC Electrical Characteristics $0°C \leq T_A \leq 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle	$V_{CC} \geq 4.5V$	4	DC	μs
Time (t_c)	$4.5V > V_{CC} \geq 2.4V$	16	DC	μs
Operating CKI	$V_{CC} \geq 4.5V$	DC	1.0	MHz
Frequency	$4.5V > V_{CC} \geq 2.4V$	DC	250	kHz
Duty Cycle (Note 4)	$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 4$ MHz external clock		60	ns
Fall Time (Note 4)	$f_1 = 4$ MHz external clock		40	ns
Instruction Cycle	$R = 30k$, $V_{CC} = 5V$			
Time using D0 as a	$C = 82$ pF	8	16	μs
RC Oscillator Dual-				
Clock Input (Note 4)				
INPUTS: (See *Fig. 3*)				
t_{SETUP}				
G Inputs	$V_{CC} \geq 4.5V$	$T_c/4 + .7$		μs
SI Input	$V_{CC} \geq 4.5V$	0.3		μs
IP Input	$V_{CC} \geq 4.5V$	1.0		μs
All Others	$V_{CC} \geq 4.5V$	1.7		μs
t_{HOLD}	$V_{CC} \geq 4.5V$	0.25		μs
	$4.5V > V_{CC} \geq 2.4V$	1.0		μs
OUTPUT PROPAGATION DELAY	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5K$			
IP7–IP0, A10–A8, SKIP t_{PD1}, t_{PD0}	$V_{CC} \geq 4.5V$		1.94	μs
	$4.5V > V_{CC} \geq 2.4V$		7.75	μs
AD/$\overline{DATA}$ t_{PD1}, t_{PD0}	$V_{CC} \geq 4.5V$		375	ns
	$4.5V > V_{CC} \geq 2.4V$		1.5	μs
ALL OTHER OUTPUTS t_{PD1}, t_{PD0}	$V_{CC} > 4.5V$		1.0	μs
	$4.5V > V_{CC} \geq 2.4V$		4.0	μs
MICROBUS TIMING Read Operation (*Fig. 4*)	$C_L = 50$ pF, $V_{CC} = 5V \pm 5\%$			
Chip select stable before $\overline{RD}$ – t_{CSR}		65		ns
Chip select hold time for $\overline{RD}$ – t_{RCS}		20		ns
$\overline{RD}$ pulse width – t_{RR}		400		ns
Data delay from $\overline{RD}$ – t_{RD}			375	ns
$\overline{RD}$ to data floating – t_{DF} (Note 4)			250	ns
Write Operation (*Fig. 5*)				
Chip select stable before $\overline{WR}$ – t_{CSW}		65		ns
Chip select hold time for $\overline{WR}$ – t_{WCS}		20		ns
$\overline{WR}$ pulse width – t_{WW}		400		ns
Data set-up time for $\overline{WR}$ – t_{DW}		320		ns
Data hold time for $\overline{WR}$ – t_{WD}		100		ns
INTR transition time from $\overline{WR}$ – t_{WI}			700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI and all other pins pulled up to V_{CC} with 20k resistors. See current drain equation on page 16.

Note 2: Test conditions: All inputs tied to V_{CC}; L lines in TRI-STATE mode and tied to Ground; all outputs tied to Ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

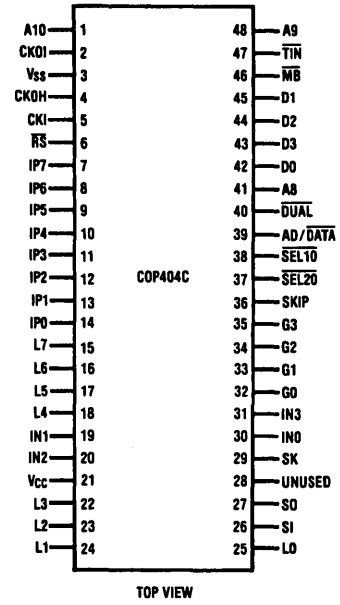
Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} to prevent entering test mode.

Note 7: $\overline{MB}$, $\overline{TIN}$, $\overline{DUAL}$, $\overline{SEL10}$, $\overline{SEL20}$, input levels at V_{CC} or V_{SS}.

Connection Diagram

Dual-In-Line Package



TL/DD/5530-2

Order Number COP404CN
See NS Package Number N48A

Pin Descriptions

Pin	Description
V_{CC}	Most positive voltage
V_{SS}	Ground
CKI	Clock input
$\overline{RS}$	Reset input
CKOI	General purpose input
L0–L7	8 TRI-STATE I/O
G0–G3	4 general purpose I/O
D1–D3	3 general purpose outputs
D0	Either general purpose output or Dual-Clock RC input
IN0–IN3	4 general purpose inputs
SO	Serial data output
SI	Serial data input
SK	Serial data clock output
IP0–IP7	I/O for ROM address and data
A8, A9, A10	3 address outputs
SKIP	Skip status output
AD/$\overline{DATA}$	Clock output
$\overline{MB}$	MICROBUS select input
CKOH	Halt I/O pin
$\overline{DUAL}$	Dual-Clock select input
$\overline{TIN}$	Timer input select pin
$\overline{SEL10}$	COP410C emulation select input
$\overline{SEL20}$	COP424C emulation select input
UNUSED	Ground

FIGURE 2

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program Memory consists of a 2048-byte external memory (typically PROM). Words of this memory may be program instructions, constants or ROM addressing data.

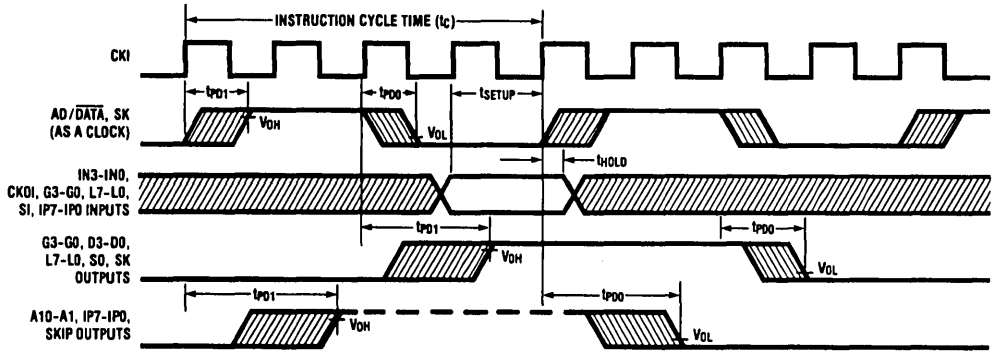
ROM addressing is accomplished by a 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

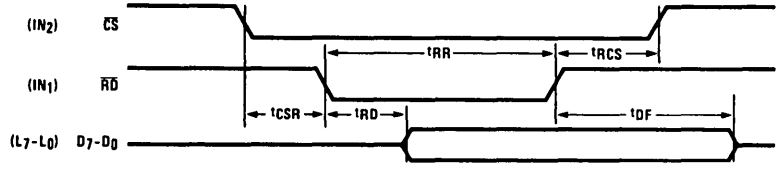
Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 × 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (B_r) select 1 of 8 data registers and lower 4 bits (B_d) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions. The B_d register also serves as a source register for 4-bit data sent directly to the D outputs.

Timing Diagrams



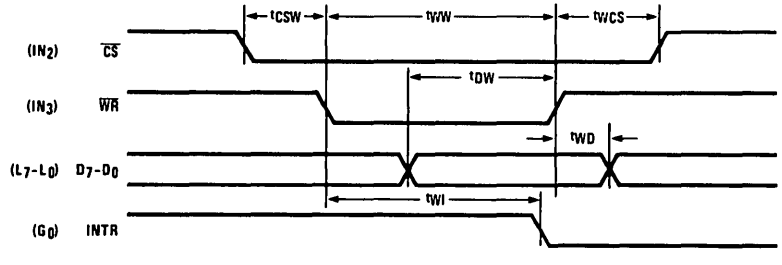
TL/DD/5530-3

FIGURE 3. Input/Output Timing



TL/DD/5530-4

FIGURE 4. MICROBUS Read Operation Timing



TL/DD/5530-5

FIGURE 5. MICROBUS Write Operation Timing

Functional Description

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the B_r and B_d portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes: as a timer if $\overline{TIN}$ pin is tied to Ground or as an external event counter if $\overline{TIN}$ pin is tied to V_{CC}. When the T counter overflows, an overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 10a*.

Four general-purpose inputs, IN3–IN0, are provided. IN1, IN2 and IN3 may be selected (by pulling $\overline{MB}$ pin low) as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in MICROBUS application.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of B_d. In the dual clock mode, D0 latch controls the clock selection (see dual oscillator below).

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G0 may be selected as an output for MICROBUS applications.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register.

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE™ I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.
3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.

b. An interrupt will be recognized only on the following conditions:
   1. EN1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN1 input.
   3. A currently executing instruction has been completed.

TABLE I. ENABLE REGISTER MODES — BITS EN0 AND EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1, SK=clock If SKL=0, SK=0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL=1, SK=clock If SKL=0, SK=0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of an ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

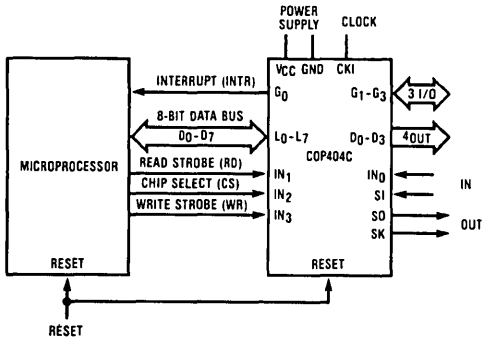
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

MICROBUS INTERFACE

With $\overline{MB}$ pin tied to Ground, the COP404C can be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN1, IN2 and IN3 general purpose inputs become MICROBUS compatible read-strobe, chip-select, and write-strobe lines, respectively. IN1 becomes $\overline{RD}$ — a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN2 becomes $\overline{CS}$ — a logic "0" on this line selects the COP404C and the μP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN3 becomes $\overline{WR}$ — a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP404C. G0 becomes INTR a "ready" output, reset by a write pulse from the μP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP404C.

This option has been designed for compatibility with National's MICROBUS - a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS National Publication). The functioning and timing relationships between the signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection of the COP404C to the MICROBUS is shown in *Figure 6*.



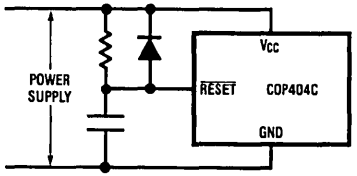
TL/DD/5530-7

FIGURE 6. MICROBUS Option Interconnect

INITIALIZATION

The external RC network shown in *Figure 7* must be connected to the $\overline{RESET}$ pin for the internal reset logic to initialize the device upon power-up. The $\overline{RESET}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{RESET}$ input, providing it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/5530-8

FIGURE 7. Power-Up Circuit

TIMER

There are two modes selected by $\overline{TIN}$ pin:

a) Time-base counter ($\overline{TIN}$ pin low). In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset. For example, using a 1MHz crystal, the instruction cycle frequency of 250 kHz (divide by 4) increments the 10-bit timer every 4 μS. By presetting the counter and detecting overflow, accurate timeouts between 16 μS (4 counts) and 4.096 mS (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

b) External event counter ($\overline{TIN}$ pin high). In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

Note: the IT instruction is not allowed in this mode.

HALT MODE

The COP404C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by two other ways (see *Figure 8*):

— Software HALT: by using the HALT instruction.

— Hardware HALT: by using the HALT I/O port CKOH. It is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing CKOH high the

Functional Description (Continued)

chip will stop as soon as CKI is high and CKOH output will stay high to keep the chip stopped if the external driver returns to high impedance state.

Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing.

The chip may be awakened by one of two different methods:

— Continue function: by forcing CKOH low, the system clock will be re-enabled and the circuit will continue to operate from the point where it was stopped. CKOH will stay low.

— Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization)

The HALT mode is the minimum power dissipation state.

Note: if the user has selected dual-clock ($\overline{\text{DUAL}}$ pin tied to Ground) AND is forcing an external clock on D0 pin AND the COP404C is running from the D0 clock, the HALT mode - either hardware or software - will NOT be entered. Thus, the user should switch to the CKI clock to HALT. Alternatively, the user may stop the D0 clock to minimize power.

Oscillator Options

There are two basic clock oscillator configurations available as shown by *Figure 9*.

— CKI oscillator: CKI is configured as a LSTTL compatible input external clock signal. The external frequency is divided by 4 to give the instruction cycle time.

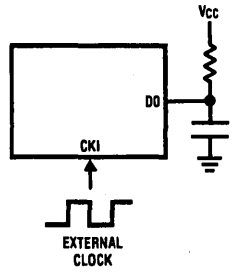
— Dual oscillator. By tying $\overline{\text{DUAL}}$ pin to Ground, pin D0 is now a single pin RC controlled Schmitt trigger oscillator input. The user may software select between the D0 oscillator (the instruction cycle time equals the D0 oscillation frequency divided by 4) by setting the D0 latch high or the CKI oscillator by resetting D0 latch low.

Note that even in dual clock mode, the counter, if used as a time-base counter, is always connected to the CKI oscillator.

For example, the user may connect up to a 1 MHz RC circuit to D0 for faster processing and a 32 kHz external clock to CKI for minimum current drain and time keeping.

Note: CTMA instruction is not allowed when the chip is running from D0 clock.

Figures 10a and *10b* show the timer and clock diagrams with and without Dual-Clock.



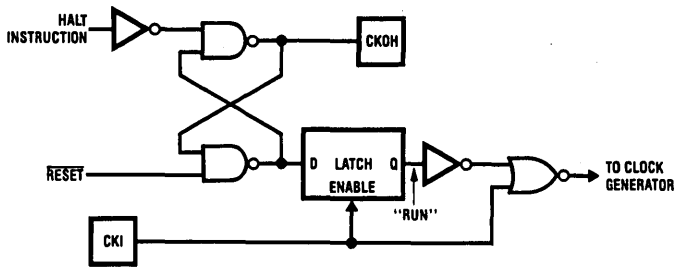
TL/DD/5530–9

R	C	Cycle Time	V_{CC}
15k	82 pF	4–9 μs	≥4.5V
30k	82 pF	8–16 μs	≥4.5V
60k	100 pF	16–32 μs	2.4–4.5V

Note: 15k≤R≤150k

50 pF≤C≤150 pF

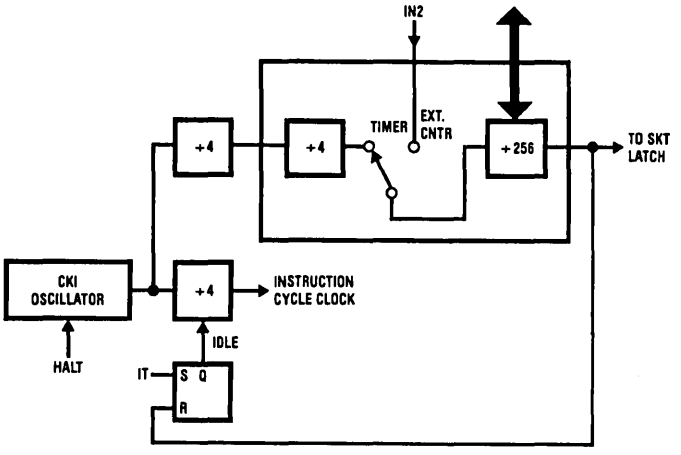
FIGURE 9. Dual-Oscillator Component Values



TL/DD/5530–10

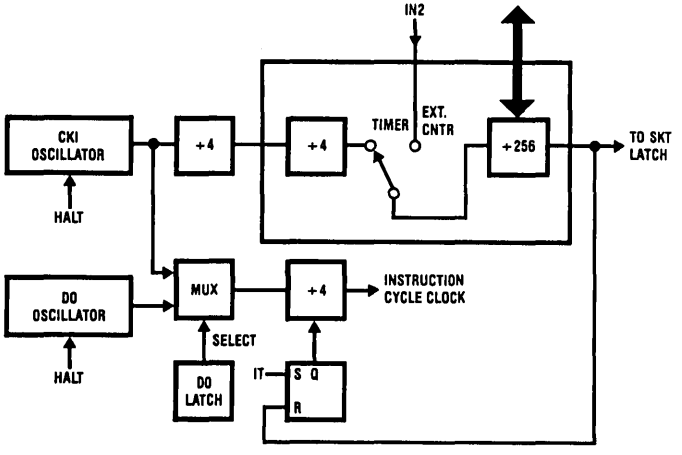
FIGURE 8. HALT Mode

Functional Description (Continued)



TL/DD/5530–11

FIGURE 10a. Clock and Timer Block Diagram without Dual-Clock



TL/DD/5530–12

Figure 10b. Clock and Timer Block Diagram with Dual-Clock

External Memory Interface

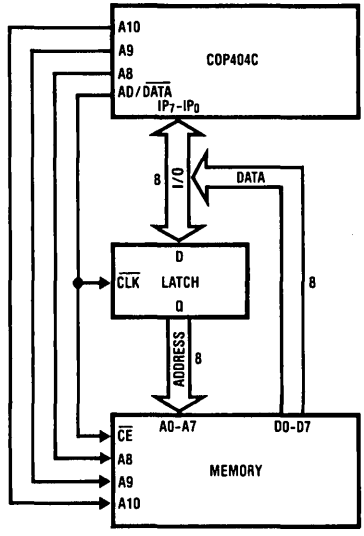
The COP404C is designed for use with an external Program Memory.

This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. LSTTL or CMOS-compatible TRI-STATE outputs
3. LSTTL or CMOS-compatible inputs
4. access time = 1.0 μs max.

Typically, these requirements are met using bipolar PROMs or MOS/CMOS PROMs, EPROMs or E²PROMs.

During operation, the address of the next instruction is sent out on A10, A9, A8 and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/DATA line; A10, A9 and A8 are dedicated address outputs, and do not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or data input. A simplified block diagram of the external memory interface is shown in *Figure 11.*



TL/DD/5530–13

FIGURE 11. External Memory Interface to COP404C

COP404C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

Table II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	7-bit RAM address register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry register
D	4-bit Data output port
EN	4-bit Enable register
G	4-bit General purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer
Instruction operand symbols	
d	4-bit operand field, 0–15 binary (RAM digit select)
r	3-bit operand field, 0–7 binary (RAM register select)
a	11-bit operand field, 0–2047
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x
Operational Symbols	
+	Plus
−	Minus
->	Replaces
<->	is exchanged with
=	Is equal to
−	
A	one's complement of A
⊕	exclusive-or
:	range of values

Instruction Set (Continued)

TABLE III. COP404C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A+RAM(B) → A	None	Add RAM to A
ADT		4A	|0011|0001|	$A+10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5–	|0101| y |	A+y → A	Carry	Add Immediate. Skip on Carry (y≠ 0)
CASC		10	|0001|0000|	$\overline{A}$+RAM(B)+C → A Carry → C	Carry	Compliment and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A} \rightarrow A$	None	Ones complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM ($PC_{10:8}$ A,M) → $PC_{7:0}$	None	Jump Indirect (note 2)
JMP	a	6– —	|0110|0|$a_{10:8}$| | $a_{7:0}$ |	a → PC	None	Jump
JP	a	—	|1| $a_{6:0}$ | (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 3)
		—	|11| $a_{5:0}$ | (all other pages)	a → $PC_{5:0}$		
JSRP	a	—	|10| $a_{5:0}$ |	PC+1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– —	|0110|1|$a_{10:8}$| | $a_{7:0}$ |	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	|0011|0011| |0011|1000|		None	HALT processor
IT		33 39	|0011|0011| |0011|1001|		None	IDLE till timer overflows then continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33 3F	|0011|0011| |0011|1111|	A → $T_{7:4}$ RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CTMA		33 2F	|0011|0011| |0010|1111|	$T_{7:44}$ → RAM(B) $T_{3:0} \rightarrow A$	None	Copy T to RAM, A
CAMQ		33 3C	|0011|0011| |0011|1100|	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	|0011|0011| |0010|1100|	$Q_{7:4}$ → RAM(B) $Q_{3:0} \rightarrow A$	None	Copy Q to RAM, A
LD	r	–5	|00| r |0101| (r=0:3)	RAM(B) → A Br⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 —	|0010|0011| |0| r | d |	RAM(r,d) → A	None	Load A with RAM pointed to direct by r,d
LQID		BF	|1011|1111|	ROM($PC_{10:8,A,M}$) → Q SB → SC	None	Load Q Indirect (Note 2)
RMB	0 1 2 3	4C 45 42 43	|0100|1100| |0100|0101| |0100|0010| |0100|0011|	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit

Instruction Set (Continued)

TABLE III. COP404C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
SMB	0	4D	|0100|1101|	1 → RAM(B)$_0$	None	Set RAM Bit
	1	47	|0100|0111|	1 → RAM(B)$_1$		
	2	46	|0100|0110|	1 → RAM(B)$_2$		
	3	4B	|0100|1011|	1 → RAM(B)$_3$		
STII	y	7–	|0111| y |	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	|00| r |0110| (r=0:3)	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	|0010|0011|	RAM(r,d) ⟷ A	None	Exchange A with RAM
		—	|1| r | d |			pointed to directly by r,d
XDS	r	–7	|00| r |0111| (r=0:3)	RAM(B) ⟷ A Bd–1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	–4	|00| r |0100| (r=0:3)	RAM(B) ⟷ A Bd+1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	—	|00| r |(d-1)| (r=0:3: d=0,9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
		33	|0011|0011|			
		—	|1| r | d | (any r, any d)			
LEI	y	33	|0011|0011|	y → EN	None	Load EN Immediate (Note 6)
		6–	|0110| y |			
XABR		12	|0001|0010|	A ⟷ Br	None	Exchange A with Br (Note 7)
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C="1"	Skip if C is True
SKE		21	|0010|0001|		A=RAM(B)	Skip if A Equals RAM
SKGZ		33	|0011|0011|		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
		21	|0010|0001|			
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|		$G_0=0$	
	1	11	|0001|0001|	2nd byte	$G_1=0$	
	2	03	|0000|0011|		$G_2=0$	
	3	13	|0001|0011|		$G_3=0$	
SKMBZ	0	01	|0000|0001|		RAM(B)$_0$=0	Skip if RAM Bit is Zero
	1	11	|0001|0001|		RAM(B)$_1$=0	
	2	03	|0000|0011|		RAM(B)$_2$=0	
	3	13	|0001|0011|		RAM(B)$_3$=0	
SKT		41	|0100|0001|		A time-base counter carry has occured since last test	Skip on Timer (Note 2)

Instruction Set (Continued)

TABLE III. COP404C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	|0011|0011|	G → A	None	Input G Ports to A
		2A	|0010|1010|			
ININ		33	|0011|0011|	IN → A	None	Input IN Inputs to A
		28	|0010|1000|			
INIL		33	|0011|0011|	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 2)
		29	|0010|1001|			
INL		33	|0011|0011|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM,A
		2E	|0010|1110|			
OBD		33	|0011|0011|	Bd → D	None	Output Bd to D Outputs
		3E	|0011|1110|			
OGI	y	33	|0011|0011|	y → G	None	Output to G Ports Immediate
		5–	|0101| y |			
OMG		33	|0011|0011|	RAM(B) → G	None	Output RAM to G Ports
		3A	|0011|1010|			
XAS		4F	|0100|1111|	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered O to N where O signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data minus 1, e.g., to load the lower four bits of B(Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 7: If $\overline{SEL2O}$ = 1, A ⟷ Br (0 → A3)

If $\overline{SEL2O}$ = 0, A ⟷ Br (0,0 → A3, A2).

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10: PC8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC (7:4), RAM(B) → PC(3:0), leaving PC(10), PC(9) and PC(8) unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10: 8, A, M. PC10, PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

Description of Selected Instructions (Continued)

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT    ; load T counter
SKT     ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is used as an external event counter ($\overline{TIN}$ pin tied to V_{CC}).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKOI and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. The state of CKOI is input into A2. A 0 is input into A1. IL latches are cleared on reset.

Instruction Set Notes

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.

c. The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, for minimum power dissipation, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. For example, an RC oscillator on D0 will draw more current than a square wave clock input since it is a slow rising signal.

If using an external square wave oscillator, the following equation can be used to calculate the COP404C operating current drain:

$$I_{co} = Iq + V \times 40 \times F_i + V \times 1400 \times F_i / 4$$

where:

I_{co} = chip operating current drain in microamps

I_q = quiescent leakage current (from curve)

F_i = CKI frequency in MegaHertz

V = chip V_{CC} in volts

For example at 5 volts V_{CC} and 400 kHz:

$$I_{co} = 20 + 5 \times 40 \times .4 + 5 \times 1400 \times .4 / 4$$

$$I_{co} = 20 + 80 + 700 = 800\ \mu A$$

at 2.4 volts V_{CC} and 30 kHz:

$$I_{co} = 6 + 2.4 \times 40 \times .03 + 2.4 \times 1400 \times .0\frac{3}{4}$$

$$I_{co} = 6 + 2.88 + 25.2 = 34.08\ \mu A$$

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$I_{ci} = I_q + V \times 40 \times F_i$$

For example, at 5 volts V_{CC} and 400 kHz

$$I_{ci} = 20 + 5 \times 40 \times .4 = 100\ \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$Ita = Ico \times \frac{To}{To+Ti} + Ici \times \frac{Ti}{To+Ti}$$

where:

I_{ta} = total average current

I_{co} = operating current

I_{ci} = idle current

T_o = operating time

T_i = idle time

I/O OPTIONS

COP404C outputs have the following configurations, illustrated in *Figure 12*.

a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL. (Used on SO, SK, AD/$\overline{DATA}$, SKIP, A10:8 and D outputs.)

b. Low Current — This is the same configuration as a. above except that the sourcing current is much less. (Used on G outputs.)

c. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control. (Used on L outputs.)

All inputs have the following configuration:

d. Input with on chip load device to V_{CC}. (Used on CKOI.)

e. HI-Z input which must be driven by the users logic. (Used on CKI, $\overline{RESET}$, IN, SI, $\overline{DUAL}$, $\overline{TIN}$, $\overline{MB}$, $\overline{SEL10}$ and $\overline{SEL20}$ inputs.)

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 13* for each of these devices to allow the designer to effectively use these I/O configurations.

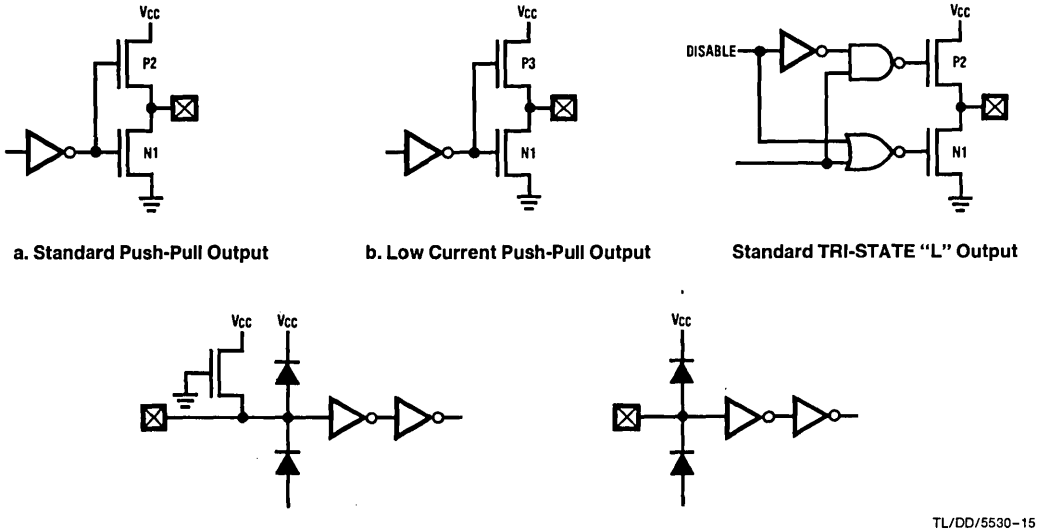


FIGURE 12. Input/Output Configurations

Typical Performance Characteristics

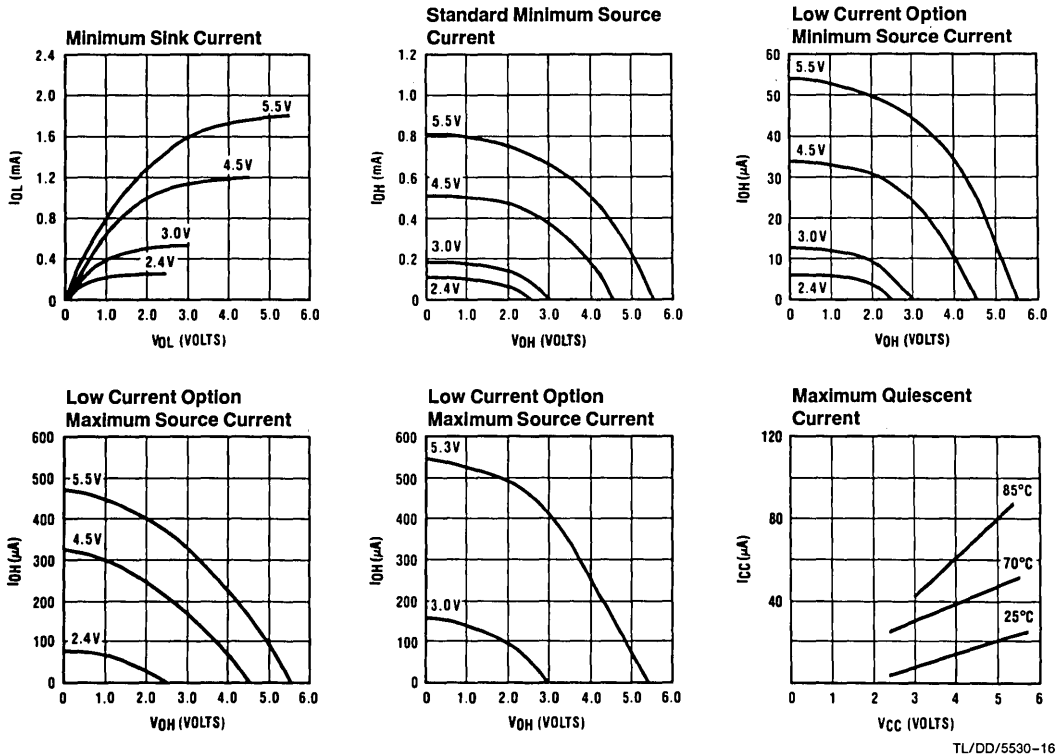
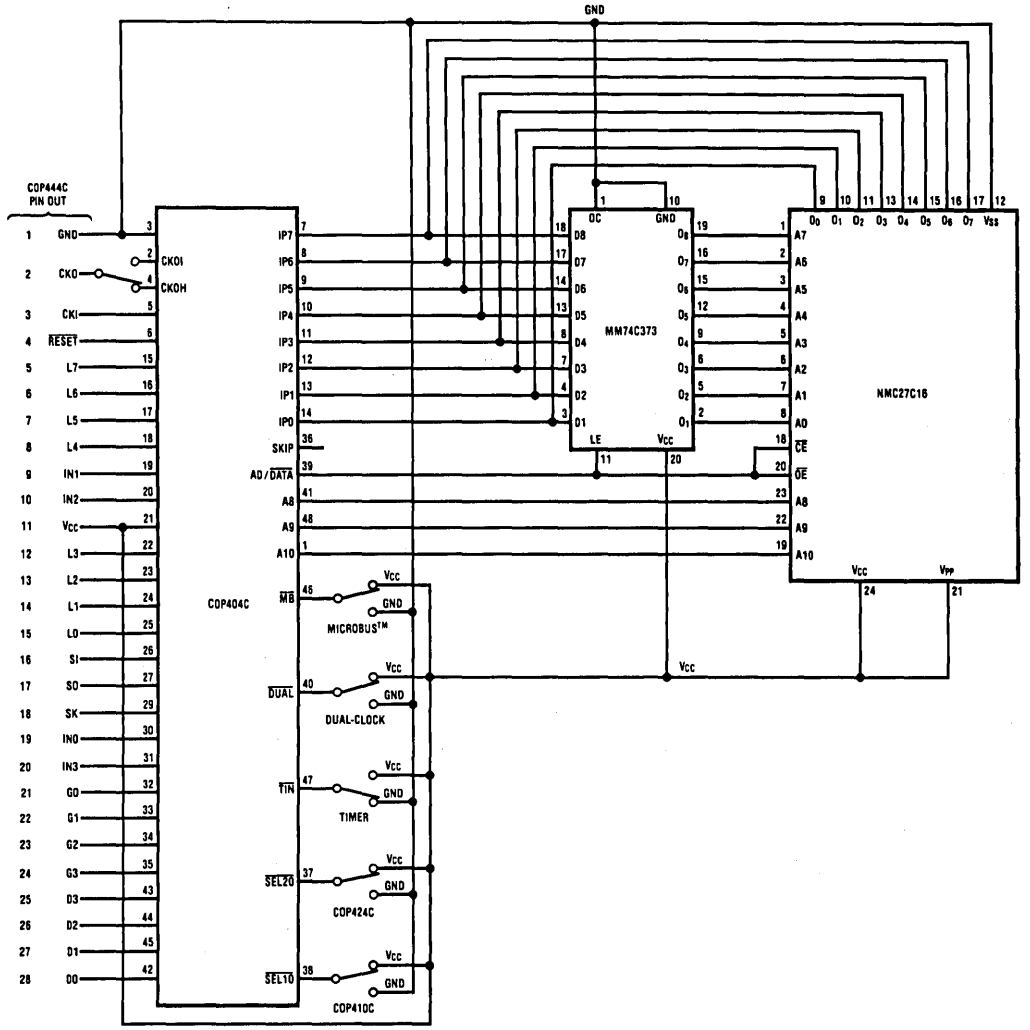


FIGURE 13. Input/Output Characteristics

Emulation

The COP404C may be used to exactly emulate the COP444C/445C, COP424C/425C, and COP410C/411C. However, the Program Counter always addresses 2k of external ROM whatever chip is being emulated. *Figure 14* shows the interconnect to implement a hardware emulation. This connection uses a NMC27C16 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.



TL/DD/5530–14

FIGURE 14. COP404C Used To Emulate A COP444C

Emulation (Continued)

When AD/$\overline{DATA}$ turns off, the EPROM is enabled and the IP7–IP0 pins will input the memory data. A10, A9 and A8 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

— CKI is divided by 4. Other divide-by are emulated by external divider.

— CKO can be emulated as a general purpose input by using CKOI or as a Halt I/O port by using CKOH.

— $\overline{MB}$ pin can be pulled low if the MICROBUS feature of the COP444C and COP424C is needed. Othewise it should be high.

— $\overline{DUAL}$ pin can be pulled low if the Dual-Clock feature of the COP444C and COP424C is needed. Otherwise it should be high.

— $\overline{TIN}$ pin controls the input of the 8-bit timer of the COP444C and COP424C (internal timer if $\overline{TIN}$ is low, external event counter if $\overline{TIN}$ is high).

— The $\overline{SEL10}$ and $\overline{SEL20}$ inputs are used to emulate the COP444C/445C, COP424C/425C, or COP410C/411C.

- When emulating the COP444C/445C, the user must configure $\overline{SEL20}=1$ and $\overline{SEL10}=1$.
- When emulating the COP424C/425C, the user must configure $\overline{SEL20}=0$ and $\overline{SEL10}=1$. In this mode, the user RAM is physically halved. As in the COP424C/425C, the user has 64 digits (256 bits) of RAM available. Pin A10 should not be connected to the program memory (most significant address bit of the program memory should be grounded if using a 2k×8 memory).
- When emulating the COP410C/411C, the user must configure $\overline{SEL20}=0$ and $\overline{SEL10}=0$. In this mode, the user has 32 digits (128 bits) of RAM available organized in the same way as the COP410C/411C - 4 registers of 8 digits each. Pins A10 and A9 should not be connected to the program memory (the 2 most significant address bits of the program memory should be grounded).

Furthermore, the subroutine stack is decreased from 3 levels to 2 levels.

The pins $\overline{SEL10}$ and $\overline{SEL20}$ change the internal logic of the device to accurately emulate the devices as indicated above. However, the user must remember that the COP424C/425C is a subset of the COP444C/COP445C with respect to memory size. The COP410C/411C is a subset both in memory size and in function. The user must take care not to use features and instructions which are not available on the COP410C/411C (see table IV. below) when using the COP404C to emulate the COP410C/411C.

TABLE IV. FEATURES AND INSTRUCTIONS NOT AVAILABLE ON COP410C/411C.

Timer	ADT		
Dual-clock	CASC		
Interrupt	CAMT		
Microbus	CTMA		
	IT		
	LDD	r, d	
	XAD	r, d	(except 3, 15)
	XABR		
	SKT		
	ININ		
	INIL		
	OGI	y	

Option Table

COP404C MASK OPTIONS

The following COP444C options have been implemented in the COP404C:

Option value	Comment
Option 1 = 0	Ground Pin — no option available
Option 2 = 1, 2	CKO is replaced by CKOI and CKOH
Option 3 = 5	CKI is external clock input divided by 4
Option 4 = 1	$\overline{RESET}$ is Hi-Z input
Option 5–8 = 0	L outputs are standard TRI-STATE
Option 9 = 1	IN1 is a Hi-Z input
Option 10 = 1	IN2 is a Hi-Z input
Option 11 = 0	V_{CC} pin — no option available
Option 12–15 = 0	L outputs are standard TRI-STATE
Option 16 = 0	SI is a Hi-Z input
Option 17 = 0	SO is a standard output
Option 18 = 0	SK is a standard output
Option 19 = 1	IN0 is a Hi-Z input
Option 20 = 1	IN3 is a Hi-Z input
Option 21–24 = 1	G outputs are low-current
Option 25–28 = 0	D outputs are standard
Option 29 = 1	No internal initialization logic
Option 30 = 0, 1	DUAL-CLOCK is pin selectable
Option 31 = 0, 1	TIMER is pin selectable
Option 32 = 0, 1	MICROBUS is pin selectable
Option 33 = N/A	48-pin package

National Semiconductor

COP404LSN-5 ROMless N-Channel Microcontrollers

General Description

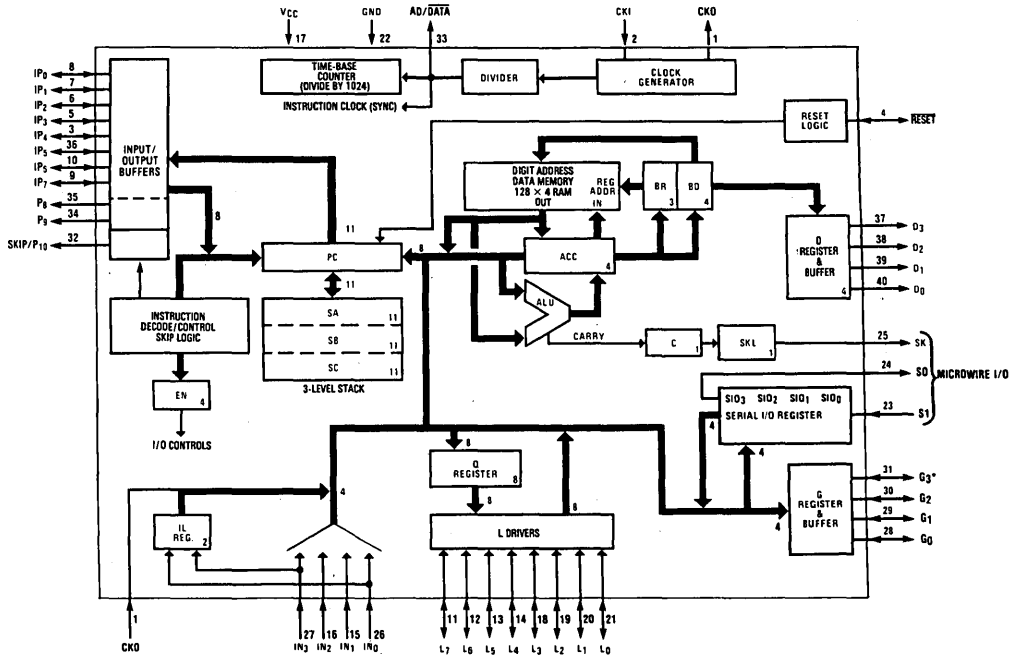
The COP404LSN-5 ROMless Microcontroller is a member of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. The COP404LSN-5 contains CPU, RAM, I/O and is identical to a COP444L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP404LSNN-5 will perform exactly as the COP444L. This important benefit facilitates development and debug of a COP program prior to masking the final part. The COP404LSN-5 is also appropriate in low volume applications, or when the program might be changing. The COP404LSN-5 may be used to emulate the COP444L, COP445L, COP420L, and the COP421L.

Use COP404LSN-5 in volume applications. For extended temperature range (−40°C to +85°C), COP304L is available on a special order basis.

Features

- Exact circuit equivalent of COP444L
- Low cost
- Powerful instruction set
- 128 x 4 RAM, addresses 2048 x 8 ROM
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–5.5V)
- Low current drain (16 mA max)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- General purpose outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family

Block Diagram



TL/DD/8817–1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	120 mA
Total Sink Current	140 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

$4.5V \le V_{CC} \le 5.5V$; $0°C \le T_A \le 70°C$

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 2)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		16	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})		2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
IP0–IP7, SI Input Levels				
Logic High	V_{CC} = 5.5V	2.4		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
All Other Inputs				
Logic High	High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
IP0–IP7, P8, P9, SKIP/P10	(Note 1)			
Logic High	I_{OH} = −80 μA	2.4		V
Logic Low	I_{OL} = 720 μA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L_0–L_7 Outputs	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
G_0–G_3 and D_0–D_3 Outputs	V_{CC} = 4.5V, V_{OL} = 1.0V	7.5		mA
CKO	V_{CC} = 4.5V, V_{OL} = 0.4V	0.2		mA
Output Source Current				
D_0–D_3, G_0–G_3 Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.0V	−30	−250	μA
SO and SK Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 1.0V	−1.2		mA
L_0–L_7 Outputs	V_{CC} = 5.5V, V_{OH} = 2.0V	−1.4	−25	mA

DC Electrical Characteristics (Continued)

$0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Input Load Source Current (I_{IL})	$V_{CC} = 5.0V$, $V_{IL} = 0V$	−10	−140	μA
Total Sink Current Allowed				
All Outputs Combined			140	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise specified

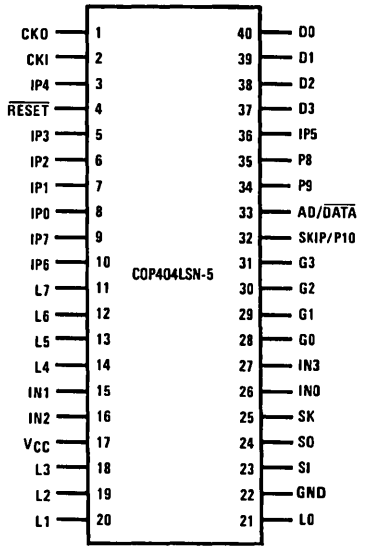
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		16	40	μs
CKI				
Input Frequency, f	(÷32 Mode)	0.8	2	MHz
Duty Cycle		30	60	%
Rise Time	$f_I = 2.0$ MHz		120	ns
Fall Time			80	ns
INPUTS:				
SI, IP7–IP0				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
IN_3–IN_0, G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
OUTPUT PROPAGATION DELAY	Test Condition: $C_L = 50$ pF, $V_{OUT} = 1.5V$			
SO, SK Outputs	$R_L = 20$ kΩ			
t_{pd1}, t_{pd0}			4.0	μs
D_3–D_0, G_3–G_0, L_7–L_0	$R_L = 20$ kΩ			
t_{pd1}, t_{pd0}			5.6	μs
IP7–IP0, P8, P9, SKIP	$R_L = 5$ kΩ			
t_{pd1}, t_{pd0}			7.2	μs
P10	$R_L = 5$ kΩ			
t_{pd1}, t_{pd0}			6.0	μs

Note 1: COP404LSN-5 has Push-Pull drivers on these outputs.

Note 2: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Connection Diagram

Dual-In-Line Package



TL/DD/8817-2

Top View

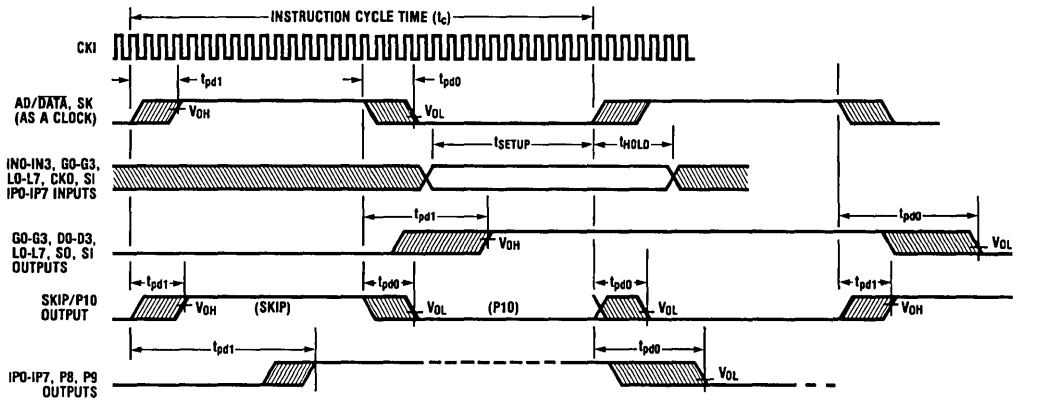
FIGURE 2

Order Number COP404LSN-5
See NS Package Number N40A

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirecitonal I/O ports with TRI-STATE®
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose outputs
SI	Serial input (or counter input
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{DATA}$	Address out/data in flag
CKI	System oscillator input
CKO	System oscillator output (COP404LSN-5)
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8, P9	2 ROM address outputs
SKIP/P10	Instruction skip output and most significant ROM address bit output

Timing Diagram



TL/DD/8817-3

FIGURE 3. Input/Output

Functional Description

A block diagram of the COP404LSN-5 is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 2048 byte external memory. As can be seen by an examination of the COP404LSN-5 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by an 11-bit PC register. Its binary value selects one of the 2048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value. Three levels of subroutine nesting are implemented by the 11-bit subroutine saves registers, SA, SB, and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

Four general-purpose inputs, IN_3–IN_0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.

Functional Description (Continued)

4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0.

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC+1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:
   1. EN_1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

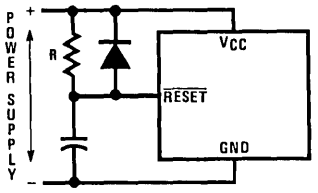
c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If the RC network is not used, the $\overline{\text{RESET}}$ pin should be left open. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/8817–4

RC ≥ 5 × Power Supply Rise Time (R > 40k)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP404LSN-5 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL-compatible inputs
4. access time = 5 μs max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P10, P9, P8, and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P9 and P8 are

Enable Register Modes — Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

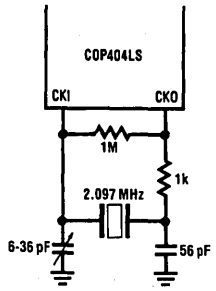
Functional Description (Continued)

dedicated address outputs, and do not need to be latched. SKIP/P10 outputs address data when AD/$\overline{DATA}$ is low. When AD/$\overline{DATA}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{DATA}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

The basic clock oscillator configurations is shown in *Figure 4*.

Crystal Controlled Oscillator—CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32.



TL/DD/8817-5

FIGURE 4. Oscillator

INPUT/OUTPUT CONFIGURATIONS

COP404LSN-5 outputs have the following configurations, illustrated in *Figure 5*:

a. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. (Used on D and G outputs.)

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application.

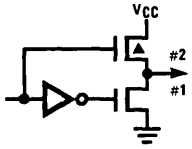
c. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

d. LED Direct Drive—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. (Used on L outputs.)

COP404LSN-5 inputs have an on-chip depletion load device to V_{CC}.

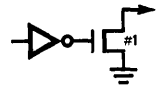
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to a logic "1".



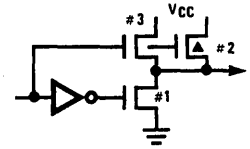
TL/DD/8817-6

a. Standard Output



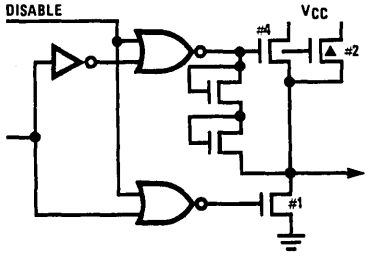
TL/DD/8817-7

b. Open-Drain Output



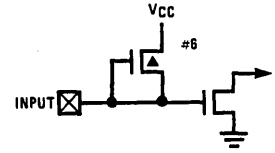
TL/DD/8817-8

c. Push-Pull Output



TL/DD/8817-9

d. L Output (LED)



TL/DD/8817-10

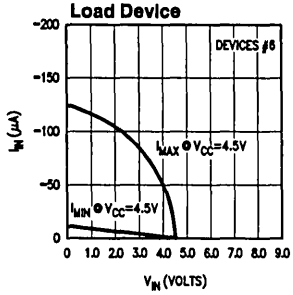
e. Input with Load

(▲ is Depletion Device)

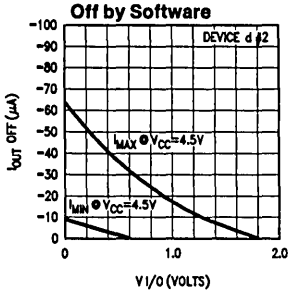
FIGURE 5. Output Configurations

Typical Performance Characteristics

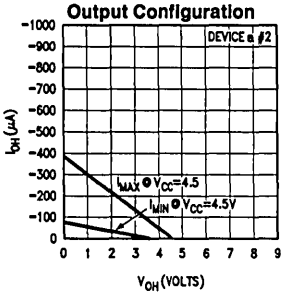
Current for Inputs with Load Device



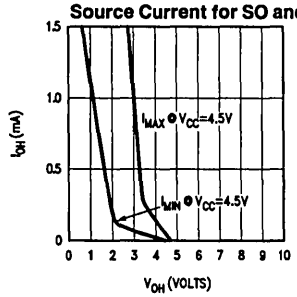
Input Current for L_0 through L_7 when Output Programmed Off by Software



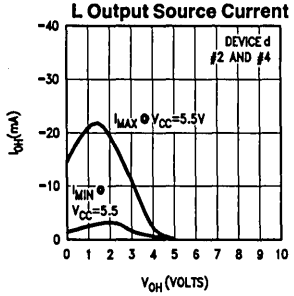
Source Current for Standard Output Configuration



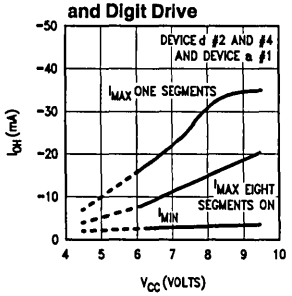
Source Current for SO and SK



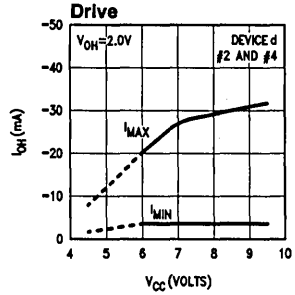
L Output Source Current



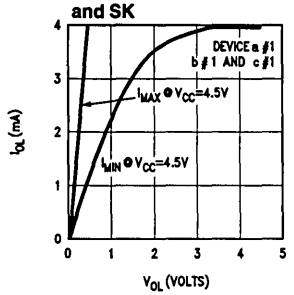
LED Output Direct Segment and Digit Drive



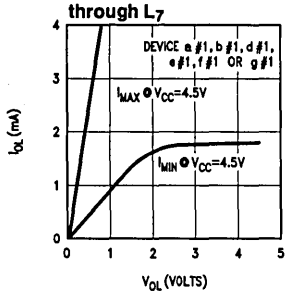
LED Output Drive Segment Drive



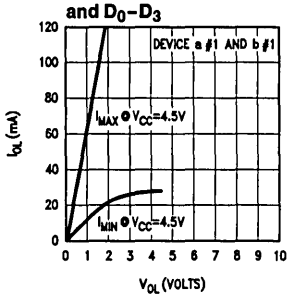
Output Sink Current for SO and SK



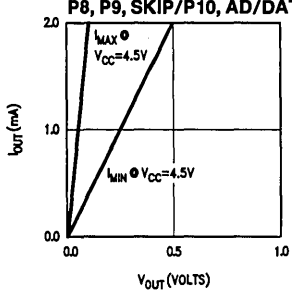
Output Sink Current for L_0 through L_7



Output Sink Current G_0–G_3 and D_0–D_3



Output Sink Current IP0–IP7, P8, P9, SKIP/P10, AD/DATA



TL/DD/8817–11

FIGURE 6. COP404LSN-5 I/O Characteristics

COP404LSN-5 Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP404LSN-5 instruction set.

TABLE I. COP404LSN-5 Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	10-bit RAM Address Register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
IP	8-bit bidirectional ROM address and Data Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
P	3-bit ROM Address Register Port
PC	11-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	3-bit Operand Field, 0–7 binary (RAM Register Select)
a	11-bit Operand Field, 0–2047 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE II. COP404LSN-5 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
ADT		4A	|0100|1010|	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	|0001|0000|	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{10:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6–	0110 0 $a_{10:8}$	a → PC	None	Jump
		--	$a_{7:0}$			
JP	a	--	1 $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 4)
			or			
		--	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	--	10 $a_{5:0}$	PC + 1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6–	0110 1 $a_{10:8}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
		--	$a_{7:0}$			
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
		3C	0011 1100			
CQMA		33	0011 0011	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
		2C	0010 1100			
LD	r	-5	00 r 0101 (r = 0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23	0010 0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		--	0 r d			
LQID		BF	1011 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	0 → $RAM(B)_1$		
	2	42	0100 0010	0 → $RAM(B)_2$		
	3	43	0100 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 0111	1 → $RAM(B)_1$		
	2	46	0100 0110	1 → $RAM(B)_2$		
	3	4B	0100 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110 (r = 0:3)	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23	0010 0011	RAM(r,d) ⟷ A	None	Exchange A with RAM pointed to directly by (r,d)
		--	1 r d			
XDS	r	-7	00 r 0111 (r = 0:3)	RAM(B) ⟷ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XIS	r	–4	|00|r|0100| (r = 0:3)	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	– –	|00|r|(d−1)| (r = 0:3; d = 0, 9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33	|0011|0011|			
		– –	|1| r | d | (any r, any d)			
LEI	y	33	|0011|0011|	y → EN	None	Load EN Immediate (Note 7)
		6–	|0110| y |			
XABR		12	|0001|0010|	A ⟷ Br (0 → A_3)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	|0011|0011|		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
		21	|0010|0001|			
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	G_0 = 0	
	1	11	|0001|0001|		G_1 = 0	
	2	03	|0000|0011|		G_2 = 0	
	3	13	|0001|0011|		G_3 = 0	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1$ = 0	
	2	03	|0000|0011|		$RAM(B)_2$ = 0	
	3	13	|0001|0011|		$RAM(B)_3$ = 0	
SKT		41	|0100|0001|		A time-base counter carry has occurred since last test	Skip on Timer (Note 2)
INPUT/OUTPUT INSTRUCTIONS						
ING		33	|0011|0011|	G → A	None	Input G Ports to A
		2A	|0010|1010|			
ININ		33	|0011|0011|	IN → A	None	Input IN Inputs to A
		28	|0010|1000|			
INIL		33	|0011|0011|	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 2)
		29	|0010|1001|			
INL		33	|0011|0011|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
		2E	|0010|1110|			
OBD		33	|0011|0011|	Bd → D	None	Output Bd to D Outputs
		3E	|0011|1110|			

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS (Continued)						
OGI	y	33 5–	0011 | 0011 0101 | y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ⟷ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds to the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selection Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP404LSN-5 programs.

XAS INSTRUCTIONS

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, $PC_{10:8}$, A, M. PC_{10}, PC_9 and PC_8 are not affected by this instruction.

Note: JID requires 2 instruction cycles to execute.

INIL INSTRUCTION

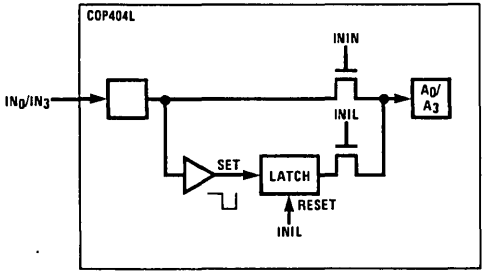
INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 7*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. INIL will input "1" into A2 on the COP404LSN-5. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC_{10}, PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_{10}, PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC).

Note: LQID takes two instruction cycle times to execute.



TL/DD/8817–12

FIGURE 7. INIL Hardware Implementation

Description of Selected Instructions (Continued)

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP404LSN-5 to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz oscillator as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷ 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP404LSN-5 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23 or 27 will access data in the next group of four pages.

Typical Applications

PROM-BASED SYSTEM

The COP404LSN-5 may be used to exactly emulate the COP444L. *Figure 8* shows the interconnect to implement a COP444L hardware emulation. This connection uses a MM2716 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74LS373) latches the addresses to drive the memory.

When AD/$\overline{\text{DATA}}$ turns off, the EPROM is enabled and the IP7–IP0 pins will input the memory data. P8, P9 and SKIP/P10 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

The other 28 pins of the COP404LSN-5 may be configured exactly the same as a COP444L. The COP404LSN-5 V_{CC} can vary from 4.5V to 5.5V. However, 5V is used for the memory.

For In-Circuit emulation, see also COP444LP.

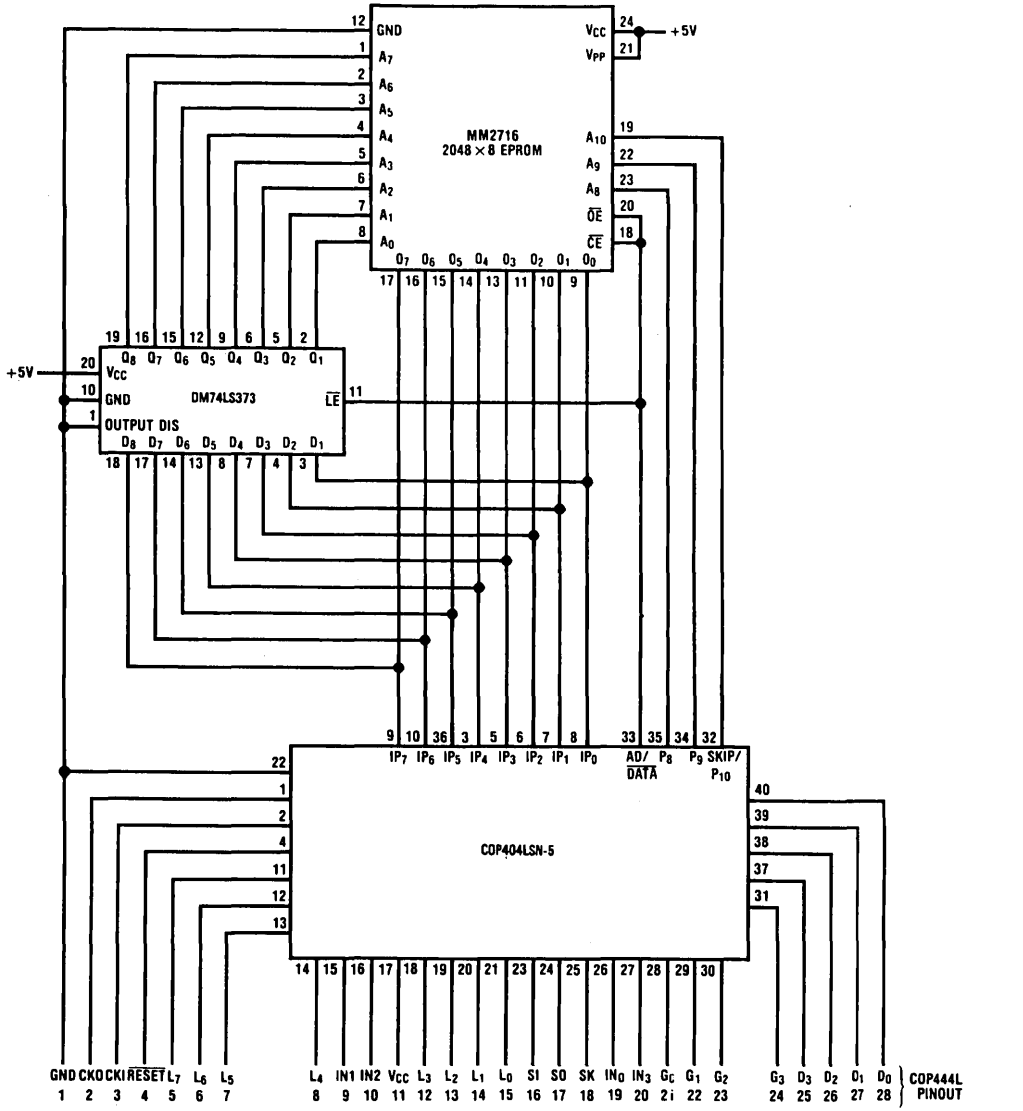
COP404LSN-5 Mask Options

The following COP444L options have been implemented on the COP404LSN-5.

Option Value	Comment
Option 1 = 0	Ground, no option available
Option 2 = 0	CKO is clock generator output to crystal/resonator
Option 3 = 0	CKI is oscillator input (divide by 32)
Option 4 = 0	$\overline{\text{RESET}}$ pin has load device to V_{CC}
Option 5 = 2	L_7
Option 6 = 2	L_6 } have LED direct-drive
Option 7 = 2	L_5 } output
Option 8 = 2	L_4
Option 9 = 0	IN1 has load device to V_{CC}
Option 10 = 0	IN2 has load device to V_{CC}
Option 11 = 1	V_{CC} 4.5V to 5.5V operation
Option 12 = 2	L_3
Option 13 = 2	L_2 } have LED direct-drive
Option 14 = 2	L_1 } output
Option 15 = 2	L_0
Option 16 = 0	SI has load to V_{CC}
Option 17 = 2	SO has push-pull output

Option Value	Comment
Option 18 = 2	SK has push-pull output
Option 19 = 0	IN0 has load device to V_{CC}
Option 20 = 0	IN3 has load device to V_{CC}
Option 21 = 0	G_0
Option 22 = 0	G_1 } have high current
Option 23 = 0	G_2 } standard output
Option 24 = 0	G_3
Option 25 = 0	D_3
Option 26 = 0	D_2 } have high current
Option 27 = 0	D_1 } standard output
Option 28 = 0	D_0
Option 29 = 1	L
Option 30 = 1	IN } have higher voltage
Option 31 = 1	G } input levels
Option 32 = 0	SI has standard input level
Option 33 = 0	$\overline{\text{RESET}}$ has Schmitt trigger input
Option 34 = 0	CKO has standard input levels
Option 35 = N/A	40-pin package

Typical Applications (Continued)



TL/DD/8817–13

FIGURE 8. COP404LSN-5 System Diagram

National Semiconductor

ADVANCE INFORMATION

COP420P/COP444CP/COP444LP Piggyback EPROM Microcontrollers

General Description

The COP420P, COP444CP, and COP444LP are piggyback versions of the COPS™ microcontroller families. These devices are identical to their respective device except the program ROM has been removed. The device package incorporates the circuitry and socket on top of package to accommodate the piggyback EPROM—MM2716, NMC27C16 or other appropriate EPROMs. With the addition of an EPROM the device performs exactly as its masked equivalent.

The device is a complete microcontroller system with CPU, RAM, I/O and EPROM socket in a 28-lead package. The completed package allows field test of the system in the final electrical and mechanical configuration. This important benefit facilitates development and debug of the COP400 program prior to masking of a production part.

These devices are also economical in low and medium volume applications or when the program may require changing.

Device Selection	Device Emulated	Piggyback Device
Low Power NMOS	COP420L, COP444L	COP444LP
High Speed NMOS	COP420	COP420P
Low Power CMOS	COP424C, COP444C	COP444CP

Features

COP444LP

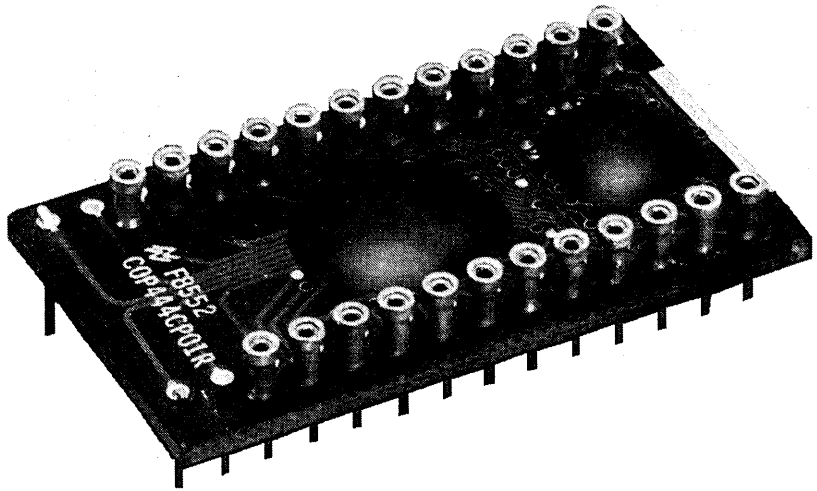
- 16 μs instruction time
- Same Specification as COP404LSN-5

COP420P

- 4 μs instruction time
- Same Specification as COP402N

COP444CP

- 4 μs instruction time
- Fully static (can turn off clock)
- Power-saving IDLE state and Halt mode
- Same Specification as COP404CN



TL/DD/8705–10

Section 2

COP800 Family

Section 2 Contents

National Semiconductor

The 8-Bit COP800 Family: Optimized for Value

National's COP800 family provides cost-effective solutions for feature-rich, 8-bit microcontroller applications.

Key Features

- High-performance 8-bit microcontroller
- Full 8-bit architecture and implementation
- 1 μs instruction-cycle time
- High code efficiency with single-byte, multiple-function instructions
- UART
- A/D converter
- Watchdog logic monitor
- On-chip ROM to 4 kbytes
- On-chip RAM to 192 bytes
- EEPROM
- M²CMOS™ fabrication
- MICROWIRE/PLUS™ serial interface
- ROMless versions available
- Wide operating voltage range: +2.5V to +6V
- Military temp range available: −55°C to +125°C
- MIL-STD-883C versions available
- 20- to 44-pin packages

The COP800 combines a powerful single-byte, multiple-function instruction set with a memory-mapped core architecture similar to the HPC™.

And like the HPC, the COP800 family supports a wide variety of ROM, RAM, I/O and peripheral functions.

The COP800 has an instruction-cycle time of only 1 μs, and because over 70% of its instruction set is composed of single-cycle, single-byte instructions, the COP800 can deliver exceptional performance for an 8-bit engine.

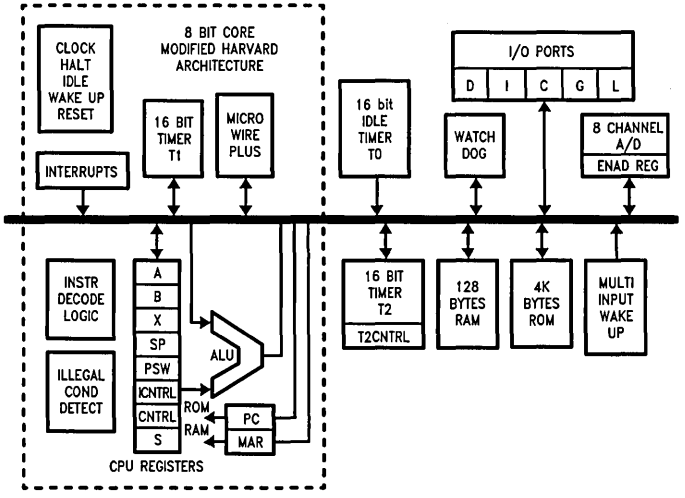
And since it's fabricated in National's advanced M²CMOS process, the COP800 has low current drain, low heat dissipation, and a wide operating voltage range.

Key Applications

- Automotive systems
- Process control
- Robotics
- Telecommunications
- AC-motor control
- DC-motor control
- Keyboard controllers
- Modems
- RS232C controllers

The COP800 family offers high performance in a low-cost, easy-to-design-in package.

COP888CF Block Diagram



TL/XX/0073–3

2

COP800 Family of Microcontrollers

Commercial Temp Version 0°C to +70°C	Industrial Temp Version −40°C to +85°C	Military Temp Version −55°C to +125°C	Memory		Features						
			ROM (Bytes)	RAM (Bytes)	I/O		Interrupt	Stack	Timer Base Counters	Size (Pins)	Other
					I/O Pins	Serial I/O					
	COP820C	COP620C	1.0k	64	24	Yes	3 Sources	In RAM	1	28	
	COP821C	COP621C	1.0k	64	20	Yes	3 Sources	In RAM	1	24	
	COP822C	COP622C	1.0k	64	16	Yes	3 Sources	In RAM	1	20	
	COP8720C		1.0k EE	64	24	Yes	3 Sources	In RAM	1	28	64 x 8 EEPROM in RAM
	COP8721C		1.0k EE	64	20	Yes	3 Sources	In RAM	1	24	64 x 8 EEPROM in RAM
	COP8722C		1.0k EE	64	16	Yes	3 Sources	In RAM	1	20	64 x 8 EEPROM in RAM
	COP840C	COP640C	2.0k	128	24	Yes	3 Sources	In RAM	1	28	
	COP841C	COP641C	2.0k	128	20	Yes	3 Sources	In RAM	1	24	
	COP842C	COP642C	2.0k	128	16	Yes	3 Sources	In RAM	1	20	
	COP884CF	COP684CF	4.0k	128	21	Yes	10 Sources	In RAM	2	28	2 PWM & A/D
	COP884CG	COP684CG	4.0k	192	23	Yes	12 Sources	In RAM	3	28	3 PWM & UART
	COP884CL	COP684CL	4.0k	128	23	Yes	10 Sources	In RAM	2	28	2 PWM
	COP888CF	COP688CF	4.0k	128	33/37	Yes	10 Sources	In RAM	2	40/44	2 PWM & A/D
	COP888CG	COP688CG	4.0k	192	35/39	Yes	12 Sources	In RAM	3	40/44	3 PWM & UART
	COP888CL	COP688CL	4.0k	128	33/39	Yes	10 Sources	In RAM	2	40/44	2 PWM

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPS™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

Development Support (Continued)

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	TBD	Programmer's Manual		TBD

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains

DIAL-A-HELPER Users Manual

Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

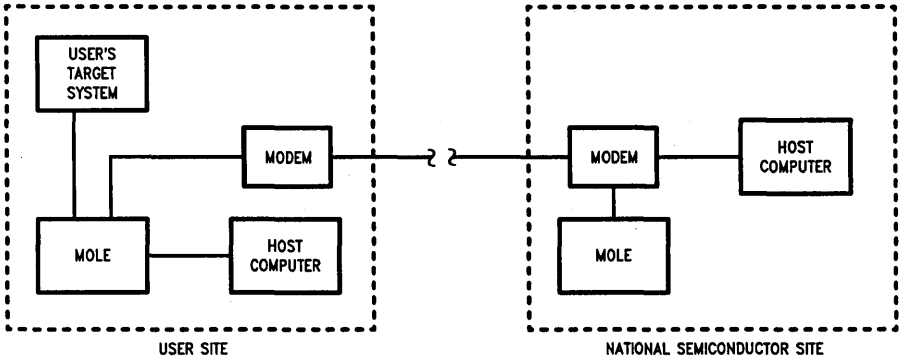
The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Development Support (Continued)

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Set-Up: Length: 8-bit
Parity: none
Stop Bit: 1
Operation: 24 hrs., 7 days

DIAL-A-HELPER



TL/XX/0073-2

National Semiconductor

PRELIMINARY

COP620C/COP621C/COP622C/COP640C/COP641C/COP642C/COP820C/COP821C/COP822C/COP840C/COP841C/COP842C Single-Chip microCMOS Microcontrollers

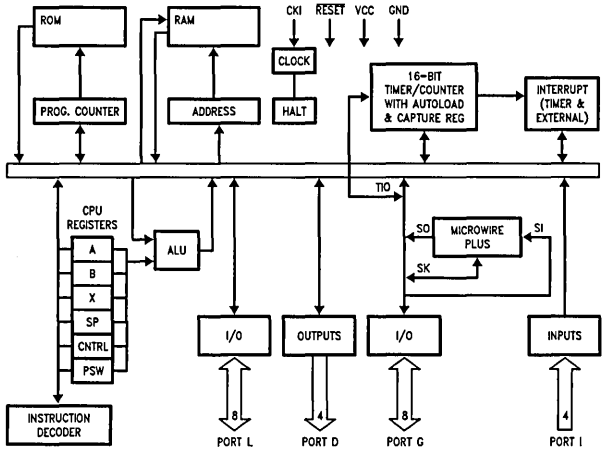
General Description

The COP820C and COP840C are members of the COPS™ microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP820C and COP840C to the specific application. The part operates over a voltage range of 2.5 to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate. The part may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Features

- Low Cost 8-bit microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain (2.2 mA at 3 μs instruction rate)
  Low current static HALT mode (Typically < 1 μA)
- Single supply operation: 2.5 to 6.0V
- 1024 bytes ROM/64 Bytes RAM—COP820C
- 2048 bytes ROM/128 Bytes RAM—COP840C
- 16-bit read/write timer operates in a variety of modes
  — Timer with 16-bit auto reload register
  — 16-bit external event counter
  — Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  — Reset master clear
  — External interrupt with selectable edge
  — Timer interrupt or capture interrupt
  — Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instruction single byte
- BCD arithmetic instructions
- MICROWIRE PLUS™ serial I/O
- 28 pin package (optionally 24 or 20 pin package)
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Extended temperature ranges: −40°C to +85°C (−55°C to +125°C to be available)
- ROMless mode for accurate emulation and external program capability—expandable to 32k bytes in ROM-less mode
- Form, fit and function EEPROM emulation device (COP8720C)
- Piggyback emulation devices (COP820CP/COP840CP)
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/9103-1

FIGURE 1

COP820C/COP821C/COP822C/COP840C/COP841C/COP842C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	50 mA
Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	−65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		2.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current (see page 17)					
High Speed Mode, CKI = 20 MHz	V_{CC} = 6V, tc = 1 μs			9	mA
Normal Mode, CKI = 5 MHz	V_{CC} = 6V, tc = 2 μs			4	mA
Normal Mode, CKI = 2 MHz	V_{CC} = 2.5V, tc = 5 μs			0.7	mA
(Note 2)					
HALT Current	V_{CC} = 6V, CKI = 0 MHz		<1	10	μA
(Note 3)					
Input Levels					
$\overline{\text{RESET}}$, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6.0V, V_{IN} = 0V	−2		+2	μA
Input Pullup Current	V_{CC} = 6.0V, V_{IN} = 0V	40		250	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	2			mA
All Others					
Source (Weak Pull-Up)	V_{CC} = 4.5V, V_{OH} = 3.2V	10		110	μA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	μA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			
TRI-STATE Leakage		−2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 5) Without Latchup (Room Temp)				±100	mA
RAM Retention Voltage, Vr	500 ns Rise and Fall Time (Min)		2.0		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports TRI-STATED and tied to ground, all outputs low and tied to ground.

Note 4: Human body mode, 100 pF through 1500Ω.

Note 5: Except pins 3, 4, 24
pins 3, 24 +60 mA, −100 mA
pin 4 +100 mA, −25 mA

AC Electrical Characteristics $-40°C < T_A < +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$	1		DC	μs
(Div-by 20)	$2.5V \leq V_{CC} < 4.5V$	2.5		DC	μs
Normal Mode	$V_{CC} \geq 4.5V$	2		DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$	5		DC	μs
R/C Oscillator Mode	$V_{CC} \geq 4.5V$	3		DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 6)	fr = Max (÷20 Mode)	33		66	%
Rise Time (Note 6)	fr = 20 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 20 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	200			ns
	$2.5V \leq V_{CC} < 4.5V$	500			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	60			ns
	$2.5V \leq V_{CC} < 4.5V$	150			ns
Output Propagation Delay	C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$			0.7	μs
	$2.5V \leq V_{CC} < 4.5V$			1.75	μs
All Others	$V_{CC} \geq 4.5V$			1	μs
	$2.5V \leq V_{CC} < 4.5V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UV})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_C			
Interrupt Input Low Time		t_C			
Timer Input High Time		t_C			
Timer Input Low Time		t_C			
Reset Pulse Width		1.0			μs

Note 6: Parameter sampled but not 100% tested.

AC Electrical Characteristics in ROMless Mode $-40°C < T_A < 85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$		2	DC	μs
(Div-by 20)	$2.5V \leq V_{CC} < 4.5V$		5	DC	μs
Normal Mode	$V_{CC} \geq 4.5V$		4	DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$		10	DC	μs
R/C Oscillator Mode	$V_{CC} \geq 4.5V$		6	DC	μs
(See Page 8)	$2.5V \leq V_{CC} < 4.5V$		15	DC	μs
CKI Clock Duty Clock	fr = Max (÷20 Mode)	40		60	%
Rise Time	fr = 10 MHz Ext Clock		24		ns
Fall Time	fr = 10 MHz Ext Clock		16		ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$		400		ns
	$2.5V \leq V_{CC} < 4.5V$		800		ns
t_{HOLD}	$V_{CC} \geq 4.5V$		120		ns
	$2.5V \leq V_{CC} < 4.5V$		300		ns
Output Propagation Delay	C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$		1.4		μs
	$2.5V \leq V_{CC} < 4.5V$		3.5		μs
All Others	$V_{CC} \geq 4.5V$		2		μs
	$2.5V \leq V_{CC} < 4.5V$		5		μs
Minimum Pulse Width					
Interrupt Input		t_C			
Timer Input		t_C			
Reset Pulse Width		1.0			μs

COP620C/COP621C/COP622C/COP640C/COP641C/COP642C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	40 mA
Total Current out of GND Pin (Sink)	48 mA
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-55°C \le T_A \le +125°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current					
High Speed Mode, CKI = 18 MHz	V_{CC} = 5.5V, tc = 1.1 μs			15	mA
Normal Mode, CKI = 4.5 MHz	V_{CC} = 5.5V, tc = 2.2 μs			5	mA
(Note 2)					
HALT Current	V_{CC} = 5.5V, CKI = 0 MHz		<10		μA
(Note 3)					
Input Levels					
RESET, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 5.5V, V_{IN} = 0V	−5		+5	μA
Input Pullup Current	V_{CC} = 4.5V, V_{IN} = 0V	35		300	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.8V	0.35			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	9			mA
All Others					
Source (Weak Pull-Up)	V_{CC} = 4.5V, V_{OH} = 3.2V	9		120	μA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.8V	0.35			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.4			mA
TRI-STATE Leakage		−5.0		+5.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				12	mA
All Others				2.5	mA
Maximum Input Current (Room Temp) Without Latchup (Note 5)				±100	mA
RAM Retention Voltage, Vr	500 ns Rise and Fall Time (Min)		2.5		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Human body mode, 100 pF through 1500Ω.

Note 5: Except pins 3, 4, 24
pins 3, 24: +60 mA
pin 4: −25 mA

COP620C/COP621C/COP622C/COP640C/COP641C/COP642C

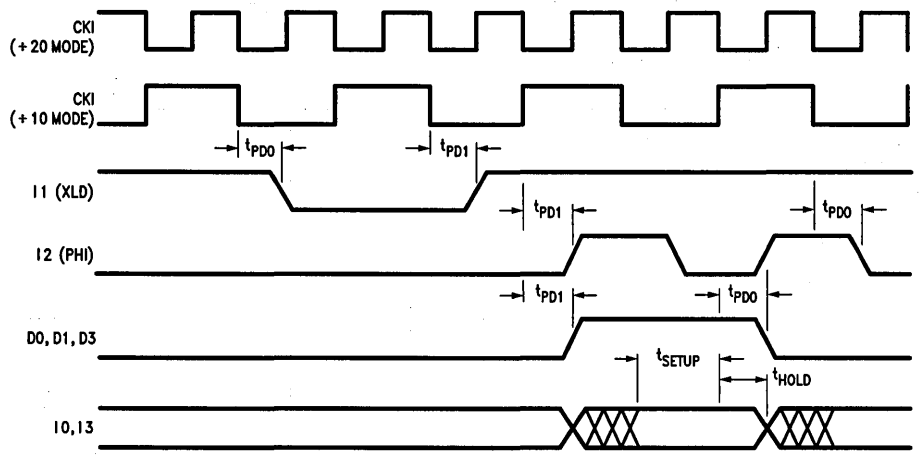
AC Electrical Characteristics $-55°C < T_A < +125°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode (Div-by 20)	$V_{CC} \geq 4.5V$	1.1		DC	μs
Normal Mode (Div-by 10)	$V_{CC} \geq 4.5V$	2.2		DC	μs
CKI Clock Duty Cycle (Note 6)	fr = Max (÷20 Mode)	33		66	%
Rise Time (Note 6)	fr = 18 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 18 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	220			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	66			ns
Output Propagation Delay t_{PD1}, t_{PD0}	$R_L = 2.2k$, $C_L = 100$ pF				
SO, SK	$V_{CC} \geq 4.5V$			0.8	μs
All Others	$V_{CC} \geq 4.5V$			1.1	μs
MICROWIRE Setup Time t_{UWS}		20			ns
MICROWIRE Hold Time t_{UWH}		56			ns
MICROWIRE Output Valid Time t_{UV}				220	ns
Input Pulse Width					
Interrupt Input High Time		tc			
Interrupt Input Low Time		tc			
Timer Input High Time		tc			
Timer Input Low Time		tc			
Reset Pulse Width		1			μs

Note 6: Parameter sampled but not 100% tested.

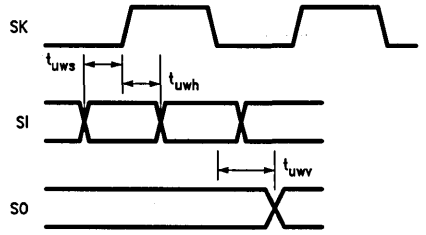
AC Electrical Characteristics in ROMless Mode $-55°C < T_A < +125°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode (Div-by 20)	$V_{CC} \geq 4.5V$		2.2	DC	μs
Normal Mode (Div-by 10)	$V_{CC} \geq 4.5V$		4.4	DC	μs
CKI Clock Duty Clock	fr = Max (÷20 Mode)	40		60	%
Rise Time	fr = 9 MHz Ext Clock		24		ns
Fall Time	fr = 9 MHz Ext Clock		16		ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$		440		ns
t_{HOLD}	$V_{CC} \geq 4.5V$		132		ns
Output Propagation Delay t_{PD1}, t_{PD0}	$R_L = 2.2k$, $C_L = 100$ pF				
SO, SK	$V_{CC} \geq 4.5V$		1.55		μs
All Others	$V_{CC} \geq 4.5V$		2.2		μs
Minimum Pulse Width					
Interrupt Input		tc			
Timer Input		tc			
Reset Pulse Width		1			μs



TL/DD/9103-2

FIGURE 2a. AC Timing Diagrams in ROMless Mode



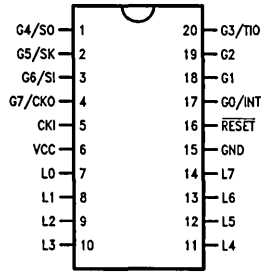
TL/DD/9103-19

FIGURE 2b. MICROWIRE/PLUS Timing

Connection Diagrams

DUAL-IN-LINE PACKAGE

20 DIP

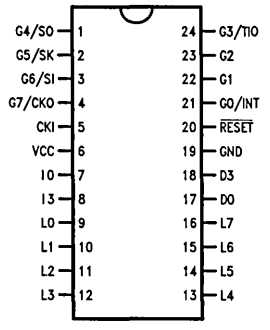


TL/DD/9103-3

Top View

Order Number COP822C-XXX/D, COP822C-XXX/N, COP842C-XXX/D or COP842C-XXX/N
See NS Package Number D20A or N20A

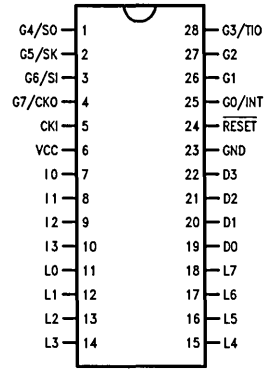
24 DIP



TL/DD/9103-4

Order Number COP821C-XXX/D, COP821C-XXX/N, COP841C-XXX/D or COP841C-XXX/N
See NS Package Number D24C or N24A

28 DIP

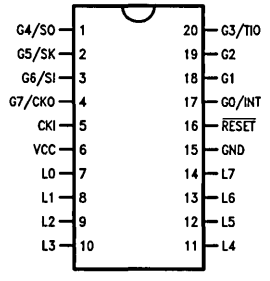


TL/DD/9103-5

Order Number COP820C-XXX/D, COP820C-XXX/N, COP840C-XXX/D or COP840C-XXX/N
See NS Package Number D28C or N28B

SURFACE MOUNT

20 SO Wide

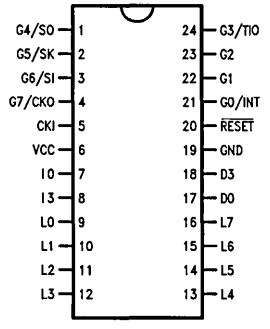


TL/DD/9103-3

Top View

Order Number COP822C-XXX/WM or COP842C-XXX/WM
See NS Package Number M20B

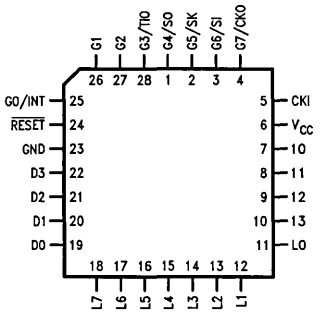
24 SO Wide



TL/DD/9103-4

Order Number COP821C-XXX/WM or COP841C-XXX/WM
See NS Package Number M24B

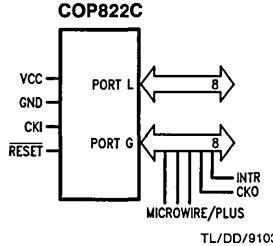
28 PLCC



TL/DD/9103-18

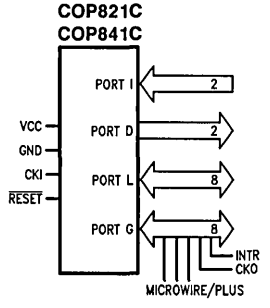
Order Number COP820C-XXX/V or COP840C-XXX/V
See NS Package Number V28A

COP822C



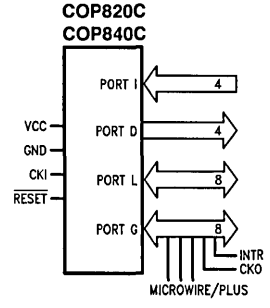
TL/DD/9103-6

COP821C
COP841C



TL/DD/9103-7

COP820C
COP840C



TL/DD/9103-8

FIGURE 3

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

$\overline{\text{RESET}}$ is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below.

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when $\overline{\text{RESET}}$ goes low.

The D2 pin is sampled at reset. If it is held low at reset the COP820C enters the ROMless mode of operation.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 15-bit Program Counter register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the program counter stack in RAM during subroutine calls and returns.

PROGRAM MEMORY

Program memory for the COP820C consists of 1024 bytes of ROM (2048 bytes of ROM for the COP840C). These bytes may hold program instructions or constant data. The program memory is addressed by the 15-bit program counter (PC). ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

The data memory address space includes on chip RAM, I/O and registers. Data memory is addressed directly by the instruction or indirectly by the B, X and SP registers.

The COP820C has 64 bytes of RAM and the COP840C has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" that can be loaded immediately, decremented or tested. Three specific registers: B, X and SP are mapped into this space, the other bytes are available for general usage.

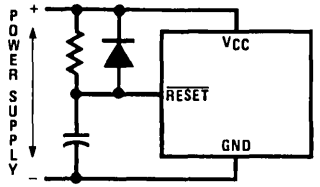
The instruction set of the COP800C permits any bit in memory to be set, reset or tested. All I/O and registers on the COP800C (except the A & PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested.

RESET

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the ports L and G are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L & G are cleared.

The external RC network shown in Figure 4 should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes. It is recommended that the components of the RC network be selected to provide a $\overline{\text{RESET}}$ delay of at least five times the power supply rise time or the minimum $\overline{\text{RESET}}$ pulse width, whichever is greater.

Functional Description (Continued)



TL/DD/9103-9

FIGURE 4. Recommended Reset Circuit

OSCILLATOR CIRCUITS

Figure 5 shows the three clock oscillator configurations available for the COP820C and COP840C.

A. CRYSTAL OSCILLATOR

The COP800C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

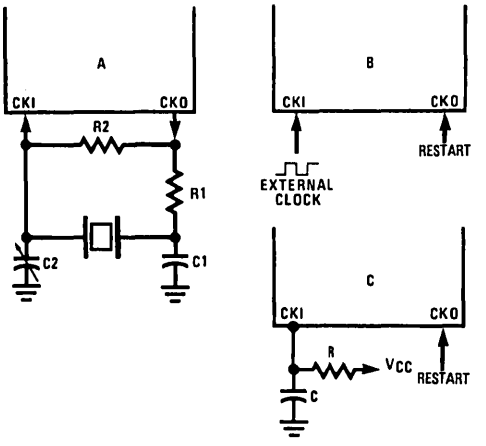
B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9103-10

FIGURE 5. Crystal and R-C Connection Diagrams

MASK OPTIONS

The COP820C and COP840C can be driven by clock inputs between DC and 20 MHz. For low input clock frequencies ($\leq$ 5 MHz) the instruction cycle frequency can be selected to be the input clock frequency divided by 10. This mode is known as the Normal Mode.

For oscillator frequencies that are greater than 5 MHz the chip must run with a divide by 20. This is known as the High Speed mode.

TABLE I. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	20	V_{CC} = 5V
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4	V_{CC} = 2.5V
0	1	200	100–150	0.455	V_{CC} = 2.5V

TABLE II. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8 to 2.2	3 to 6	V_{CC} = 5V
5.6	100	1.5 to 1.1	6 to 11	V_{CC} = 5V
6.8	100	1.1 to 0.8	7.5 to 18	V_{CC} = 2.5V

Functional Description (Continued)

The COP820C and COP840C microcontrollers have five mask options for configuring the clock input. The CKI and CKO pins are automatically configured upon selecting a particular option.

— High Speed Crystal (CKI/20) CKO for crystal configuration

— Normal Mode Crystal (CKI/10) CKO for crystal configuration

— High Speed External (CKI/20) CKO available as G7 input

— Normal Mode External (CKI/10) CKO available as G7 input

— R/C (CKI/10) CKO available as G7 input

G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

1) Oscillator operating mode—I1

2) Internal switching current—I2

3) Internal leakage current—I3

4) Output source current—I4

5) DC current caused by external input not at V_{CC} or GND—I5

Thus the total current drain, It is given as

$$It = I1 + I2 + I3 + I4 + I5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$I2 = C \times V \times f$

Where

C = equivalent capacitance of the chip.

V = operating voltage

f = CKI frequency

Some sample current drain values at V_{CC} = 6V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
20	1	9
3.58	3	2.2
2	5	1.2
0.3	33	0.2
0 (HALT)	—	<0.0001

HALT MODE

The COP820C and COP840C support a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to Vr (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the $\overline{\text{RESET}}$ or by the CKO pin. A low on the $\overline{\text{RESET}}$ line reinitializes the microcontroller and start executing from the address 0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction.

INTERRUPTS

The COP820C and COP840C have a sophisticated interrupt structure to allow easy interface to the real word. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer carry or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and continues from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Either of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

Functional Description (Continued)

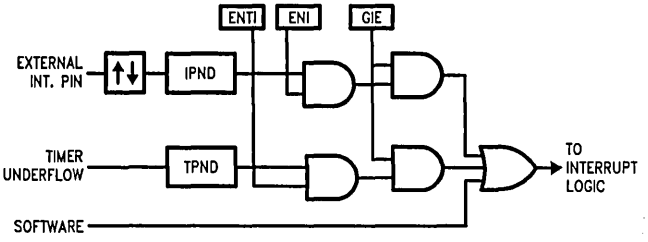


FIGURE 6. Interrupt Block Diagram

DETECTION OF ILLEGAL CONDITIONS

The COP820C and COP840C incorporate a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP820C and COP840C grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP820C and COP840C to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMS, etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 7* shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE mode. To use the MICROWIRE, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	$2t_C$
0	1	$4t_C$
1	x	$8t_C$

where,

t_C is the instruction cycle clock.

MICROWIRE PLUS OPERATION

Setting the BUSY bit in the PSW register causes the Microwire arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP820C and COP840C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 8* shows how two COP820C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP820C. The MICROWIRE Master always initiates all data exchanges. (See *Figure 8*). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summaries the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See *Figure 8*.)

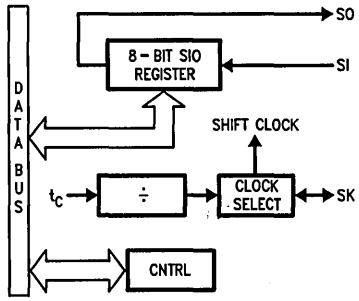
Functional Description (Continued)

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The COP820C and COP840C have a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes.



TL/DD/9103-12

FIGURE 7. MICROWIRE Block Diagram

MODE 1. TIMER WITH AUTO-LOAD REGISTER

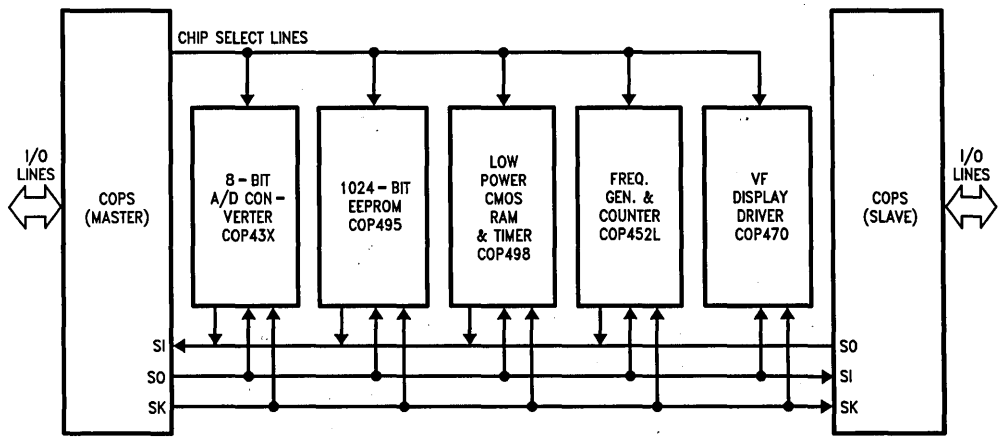
In this mode of operation the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allow the generation of square-wave outputs or pulse width modulated outputs under software control. (See *Figure 9*)

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See *Figure 9*)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See *Figure 10.*)



TL/DD/9103-13

FIGURE 8. Microwire Application

Functional Description (Continued)

TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t_C
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t_C
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t_C
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t_C

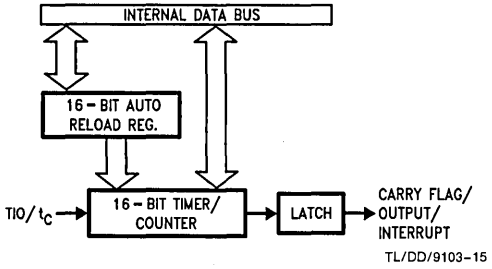


FIGURE 9. Timer/Counter Auto Reload Mode Block Diagram

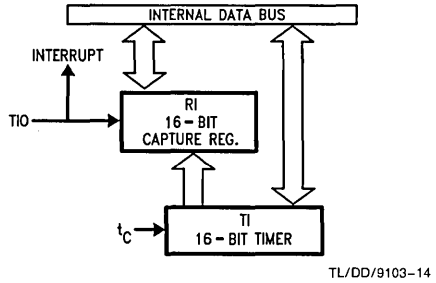


FIGURE 10. Timer Capture Mode Block Diagram

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.

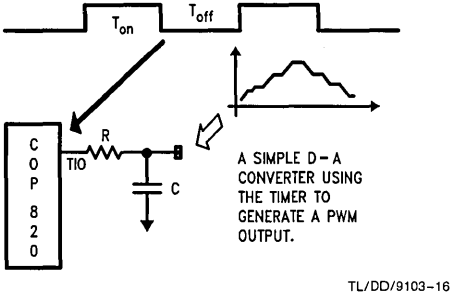


FIGURE 11. Timer Application

2

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE control register contains the following bits:

S1 & S0 Select the MICROWIRE clock divide-by

IEDG External interrupt edge polarity select (0 = rising edge, 1 = falling edge)

MSEL Enable MICROWIRE functions SO and SK

TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)

TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)

TC2 Selects the capture mode

TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0

BIT 7 BIT 0

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

GIE Global interrupt enable

ENI External interrupt enable

BUSY MICROWIRE busy shifting

IPND External interrupt pending

ENTI Timer interrupt enable

TPND Timer interrupt pending

C Carry Flag

HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE

Bit 7 Bit 0

Operating Modes

These controllers have two operating modes: Single Chip mode and the ROMless mode. The operating mode is determined by the state of the D2 pin at power on reset.

SINGLE CHIP MODE

In the Single Chip mode, the controller functions as a self contained microcontroller. It can address internal RAM and ROM. All ports configured as memory mapped I/O ports.

ROMLESS MODE

The COP820C and COP840C enter the ROMless mode of operation if the D2 pin is held at logical "0" at reset. In this case the internal ROM is disabled and the controller can now address up to 32 kbytes of external program memory. It continues to use the on board 64 bytes of RAM. The ports D and I are used to access the external program memory. By providing a serial interface to external program memory a large address space can be managed without the penalty of losing a large number of I/O pins in the process. *Figure 12* shows in schematic form the logic required for the ROMless mode operation and all support logic required to recreate the I/O.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
COP820C	
00 to 2F	On Chip RAM Bytes
30 to 7F	Unused RAM Address Space (Reads as all Ones)
COP840C	
00 to 6F	On Chip RAM Bytes
70 to 7F	Unused RAM Address Space (Reads as all Ones)
COP820C and COP840C	
80 to BF	Expansion Space for on Chip EERAM
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8–DB	Reserved for Port C
DC	Port D Data Register
DD–DF	Reserved for Port D
E0 to EF	On Chip Functions and Registers
E0–E7	Reserved for Future Parts
E8	Reserved
E9	MICROWIRE Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for COP820C and COP840C. The operand is the memory addressed by the B register or X register.

DIRECT

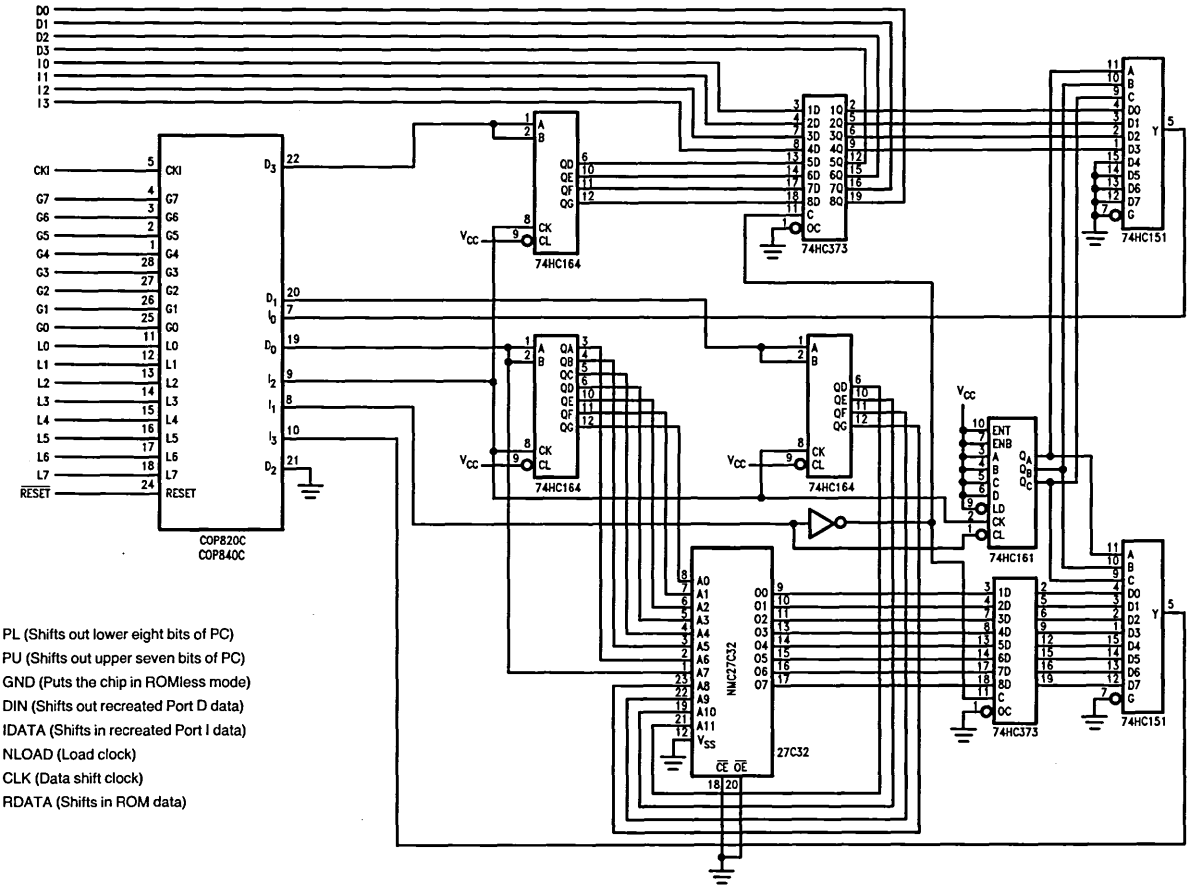
The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.



TL/DD/9103–17

FIGURE 12. COP820C and COP840C ROMless Mode Schematic

Addressing Modes (Continued)

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from −31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
MemI	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set

ADD	add	A ← A + MemI
ADC	add with carry	A ← A + MemI + C, C ← Carry HC ← Half Carry
SUBC	subtract with carry	A ← A + $\overline{\text{MemI}}$ + C, C ← Carry HC ← Half Carry
AND	Logical AND	A ← A and MemI
OR	Logical OR	A ← A or MemI
XOR	Logical Exclusive-OR	A ← A xor MemI
IFEQ	IF equal	Compare A and MemI, Do next if A = MemI
IFGT	IF greater than	Compare A and MemI, Do next if A > MemI
IFBNE	IF B not equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Decrement Reg. ,skip if zero	Reg ← Reg − 1, skip if Reg goes to 0
SBIT	Set bit	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	Reset bit	0 to bit, Mem
IFBIT	If bit	If bit, Mem is true, do next instr.
X	Exchange A with memory	A ↔ Mem
LD A	Load A with memory	A ← MemI
LD mem	Load Direct memory Immed.	Mem ← Imm
LD Reg	Load Register memory Immed.	Reg ← Imm
X	Exchange A with memory [B]	A ↔ [B] (B ← B ± 1)
X	Exchange A with memory [X]	A ↔ [X] (X ← X ± 1)
LD A	Load A with memory [B]	A ← [B] (B ← B ± 1)
LD A	Load A with memory [X]	A ← [X] (X ← X ± 1)
LD M	Load Memory Immediate	[B] ← Imm (B ← B ± 1)
CLRA	Clear A	A ← 0
INCA	Increment A	A ← A + 1
DECA	Decrement A	A ← A − 1
LAID	Load A indirect from ROM	A ← ROM(PU,A)
DCORA	DECIMAL CORRECT A	A ← BCD correction (follows ADC, SUBC)
RRCA	ROTATE A RIGHT THRU C	C → A7 → . . . → A0 → C
SWAPA	Swap nibbles of A	A7 . . . A4 ↔ A3 . . . A0
SC	Set C	C ← 1, HC ← 1
RC	Reset C	C ← 0, HC ← 0
IFC	If C	If C is true, do next instruction
IFNC	If not C	If C is not true, do next instruction
JMPL	Jump absolute long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Jump absolute	PC11..0 ← i (i = 12 bits)
JP	Jump relative short	PC ← PC + r (r is −31 to +32, not 1)
JSRL	Jump subroutine long	[SP] ← PL,[SP-1] ← PU,SP-2,PC ← ii
JSR	Jump subroutine	[SP] ← PL,[SP-1] ← PU,SP-2,PC11..0 ← i
JID	Jump indirect	PL ← ROM(PU,A)
RET	Return from subroutine	SP+2,PL ← [SP],PU ← [SP-1]
RETSK	Return and Skip	SP+2,PL ← [SP],PU ← [SP-1],Skip next instruction
RETI	Return from Interrupt	SP+2,PL ← [SP],PU ← [SP-1],GIE ← 1
INTR	Generate an interrupt	[SP] ← PL,[SP−1] ← PU,SP-2,PC ← 0FF
NOP	No operation	PC ← PC + 1

OPCODE LIST

Bits 7–4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	Bits 3–0
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0,[B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A,[B]	IFBIT 1,[B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A,[B]	IFBIT 2,[B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X−]	X A, [B−]	IFGT A, #i	IFGT A,[B]	IFBIT 3,[B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	AND A, #i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A,[B]	IFBIT 6,[B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	IFBIT 7,[B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2,[B]	RBIT 2,[B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X−]	LD A, [B−]	LD [B−], #i	DECA	SBIT 3,[B]	RBIT 3,[B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A,Md	*	SBIT 4,[B]	RBIT 4,[B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP +13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5,[B]	RBIT 5,[B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP +14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00−0EFF	JMP 0E00−0EFF	JP + 31	JP +15	E
JP -0	JP -16	LD 0FF, #1	DRSZ 0FF	*	*	*	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time (1 μs at 20 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 20 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B,Imm				1/1			(If B < 16)
LD B,Imm				2/3			(If B > 15)
LD Mem,Imm	2/2		3/3		2/2		
LD Reg,Imm				2/3			

* => Memory location addressed by B or X or directly.

Instructions Using A & C		Transfer of Control Instructions	
CLRA	1/1	JMPL	3/4
INCA	1/1	JMP	2/3
DECA	1/1	JP	1/3
LAID	1/3	JSRL	3/5
DCORA	1/1	JSR	2/5
RRCA	1/1	JID	1/3
SWAPA	1/1	RET	1/5
SC	1/1	RETSK	1/5
RC	1/1	RETI	1/5
IFC	1/1	INTR	1/7
IFNC	1/1	NOP	1/1

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC™ family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

Single Chip Emulator Device

The COP820C is fully supported by a form, fit and function emulator device, the COP8720C.

Option List

The COP820C/COP840C mask programmable options are listed out below. The options are programmed at the same time as the ROM pattern to provide the user with hardware flexibility to use a variety of oscillator configuration.

OPTION 1: CKI INPUT

= 1 Normal Mode Crystal	(CKI/10) CKO for crystal configuration
= 2 Normal Mode External	(CKI/10) CKO available as G7 input
= 3 R/C	(CKI/10) CKO available as G7 input
= 4 High Speed Crystal	(CKI/20) CKO for crystal configuration
= 5 High Speed External	(CKI/20) CKO available as G7 input

OPTION 2: COP820C/COP840C BONDING

= 1 28 pin package

= 2 24 pin package

= 3 20 pin package

The following option information is to be sent to National along with the EPROM.

Option Data

Option 1 Value_is: CKI Input

Option 2 Value_is: COP Bonding

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/ COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:

Dial-A-Helper Users Manual

Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

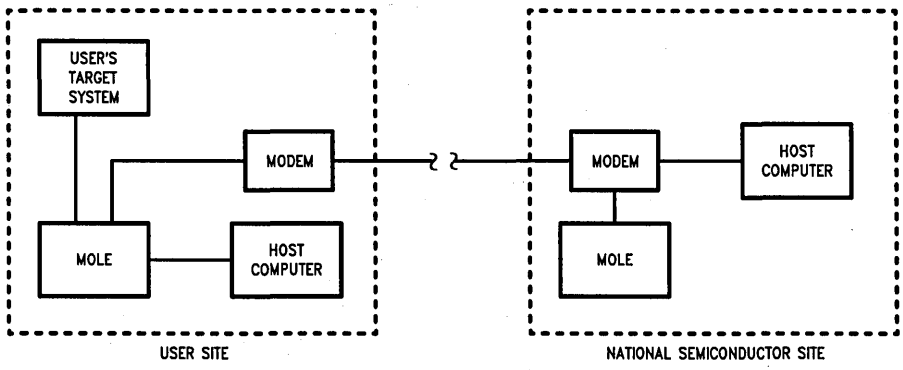
Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Setup: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs. 7 Days

DIAL-A-HELPER



TL/DD/9103–20

National Semiconductor

PRELIMINARY

COP820CP-X/COP840CP-X Piggyback EPROM Microcontrollers

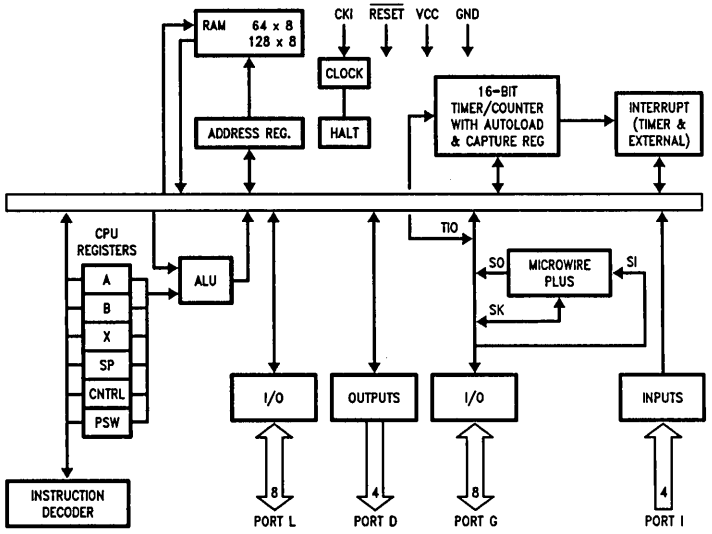
General Description

The COP820CP/COP840CP are piggyback versions of the COP820C/COP840C microcontroller families. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This microcontroller is a complete microcomputer containing all system timing, interrupt logic, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the emulator to the specific application. The part operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a 1 μs per instruction rate. The COP820CP-X/COP840CP-X are totally compatible with the ROM based COP820C/COP840 microcontroller. It serves as an economical low and medium volume emulator devices for the COP820/COP840 microcontroller family.

Features

- Low cost 8-bit CORE microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain
- Single supply operation: 4.5V to 5.5V
- Up to 32 kbytes of addressable memory
- 64 bytes of RAM (128 bytes for COP840CP)
- 16-bit read/write timer operates in a variety of modes
  - — Timer with 16-bit auto reload register
  - — 16-bit external event counter
  - — Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - — Reset master clear
  - — External interrupt with selectable edge
  - — Timer interrupt or capture interrupt
  - — Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instruction
- MICROWIRE/PLUS serial I/O
- 28 pin package
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/9683-1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
Total Current into V_{CC} Pin (Source)	150 mA
Total Current into GND Pin (Sink)	160 mA
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
High Speed Mode, CKI = 20 MHz	V_{CC} = 5.5V, t_C = 1 μs			95	mA
Normal Mode, CKI = 5 MHz	V_{CC} = 5.5V, t_C = 2 μs			90	mA
HALT Current (Note 3)	V_{CC} = 5.5V, CKI = 0 MHz (Note 4)			80	mA
INPUT LEVELS					
Reset and CKI (Crystal Osc.)					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 5.5V, V_{IN} = 0V	−10		+10	μA
G and L Port Input Hysteresis	V_{CC} = 5.5V		0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 3.2V	10		100	μA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
TRI-STATE Leakage	V_{CC} = 5.5V	−2.0		+2.0	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
RAM Retention Voltage, Vr	500 ns Rise and Fall Time (Min)		2.0		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: The rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 4: This includes the EPROM, and the pull-up resistors on the D and I ports.

Note 5: Parameter sampled but not 100% tested.

Note 6: There is one cycle delay on ports I and D.

AC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_C)					
High Speed Mode (Div-by 20)	$V_{CC} \geq 4.5V$	1		DC	μs
Normal Mode (Div-by 10)	$V_{CC} \geq 4.5V$	2		DC	μs
R/C Oscillator Mode (Div-by 10)	$V_{CC} \geq 4.5V$	3		DC	μs
CKI Clock Duty Cycle (Note 5)	fr = Max	33		66	%
Rise Time (Note 5)	fr = 20 MHz Ext Clock			12	ns
Fall Time (Note 5)	fr = 20 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	200			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	60			ns
Output Propagation Delay t_{PD1}, t_{PD0} (Note 6)	R_L = 2.2k, C_L = 100 pF				
SO, SK	$V_{CC} \geq 4.5V$			0.7	μs
All Others	$V_{CC} \geq 4.5V$			1	μs
MICROWIRE™ Setup Time (tUWS)		20			ns
MICROWIRE Hold Time (tUWH)		56			ns
MICROWIRE Output Valid Time (tUV)				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_C
Interrupt Input Low Time		1			t_C
Timer Input High Time		1			t_C
Timer Input Low Time		1			t_C
Reset Pulse Width		1			μs

Note 1: The rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 4: This includes the EPROM, and the pull-up resistors on the D and I ports.

Note 5: Parameter sampled but not 100% tested.

Note 6: There is one cycle delay on ports I and D.

EPROM Selection

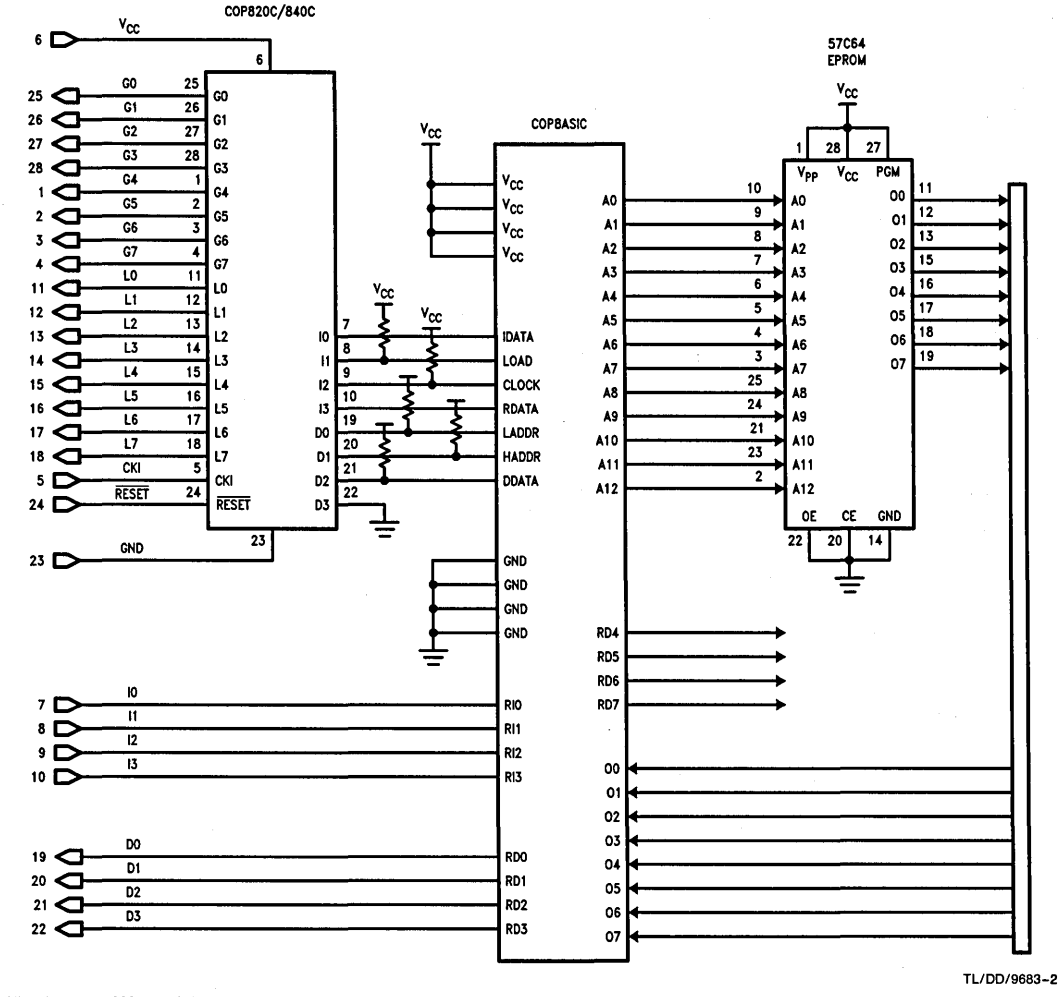
The COP820CP-X/COP840CP-X, (where X = 1, 2, 3, 4 or 5, see Table II), are the piggyback versions of the COP820C/COP840C microcontrollers. They are identical to their respective devices except that the program memory has been removed. The device package incorporates the circuitry and the socket on top of the package to allow plugging-in the EPROM 57C64, an 8 kbyte device, or any other comparable EPROM, for high speed operation. With the addition of an EPROM, these devices will perform exactly as their factory masked equivalent.

Table I lists the minimum EPROM access time for a given instruction cycle time of the microcontroller.

TABLE I

EPROM Minimum Access Time	COP Instruction Cycle Time
120 ns	1.00 μs
150 ns	1.10 μs
200 ns	1.27 μs
250 ns	1.44 μs
300 ns	1.60 μs
400 ns	1.94 μs

Connection Diagram

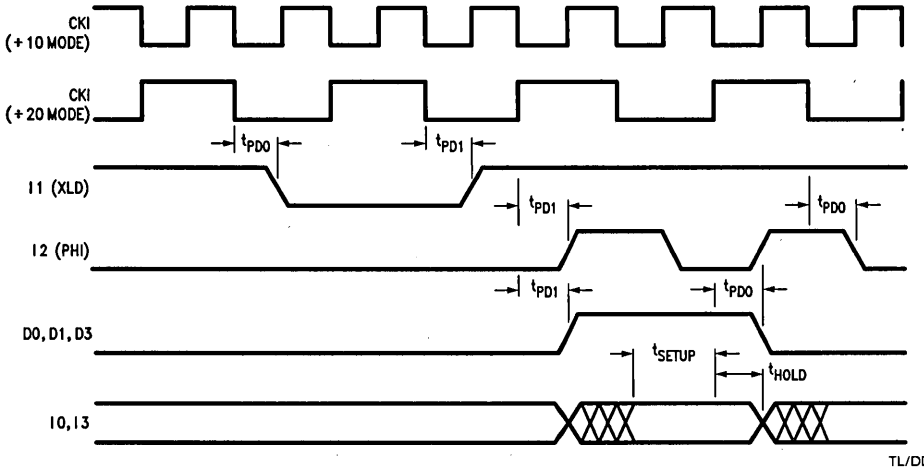


TL/DD/9683-2

All resistors are 330Ω ±20%

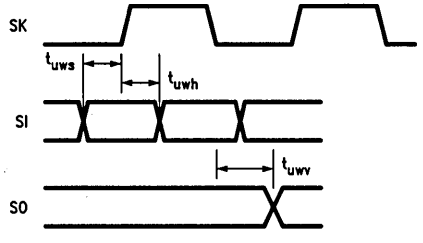
FIGURE 2

AC Timing Diagram



TL/DD/9683-3

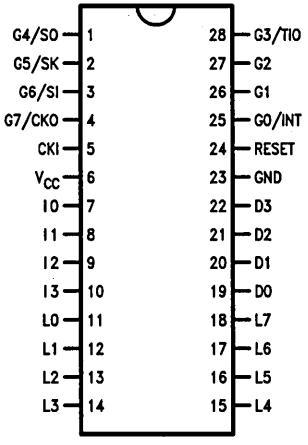
FIGURE 3



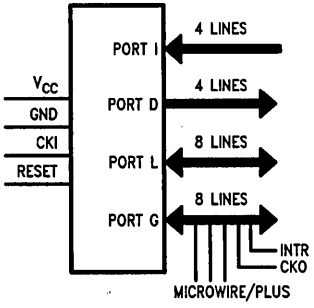
TL/DD/9683-10

FIGURE 3b

COP820CP-X/COP840CP-X Pinout Diagrams



TL/DD/9683-4



TL/DD/9683-5

Order Number COP820CP-X or COP840CP-X

FIGURE 4

Oscillator Circuits

Figure 5 shows the clock oscillator configurations available for the COP820CP-X/COP840CP-X.

A. CRYSTAL OSCILLATOR

The COP820CP-X/COP840CP-X can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table IA shows the component values required for various standard crystal values.

B. RC OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT control.

Table IB shows the variation in the oscillator frequencies as functions of the R and C component values.

TABLE IA. Crystal Oscillator Configuration, $T_A = 25°C$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	20	$V_{CC} = 5V$
0	1	30	30–36	10	$V_{CC} = 5V$
0	1	30	30	4	$V_{CC} = 5V$

TABLE IB. RC Oscillator Configuration, $T_A = 25°C$

R (kΩ)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8–2.2	3 to 6	$V_{CC} = 5V$
5.6	100	1.5–1.1	6 to 11	$V_{CC} = 5V$

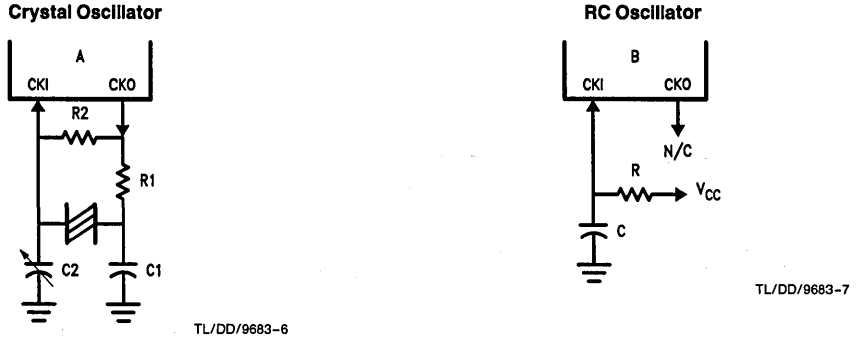
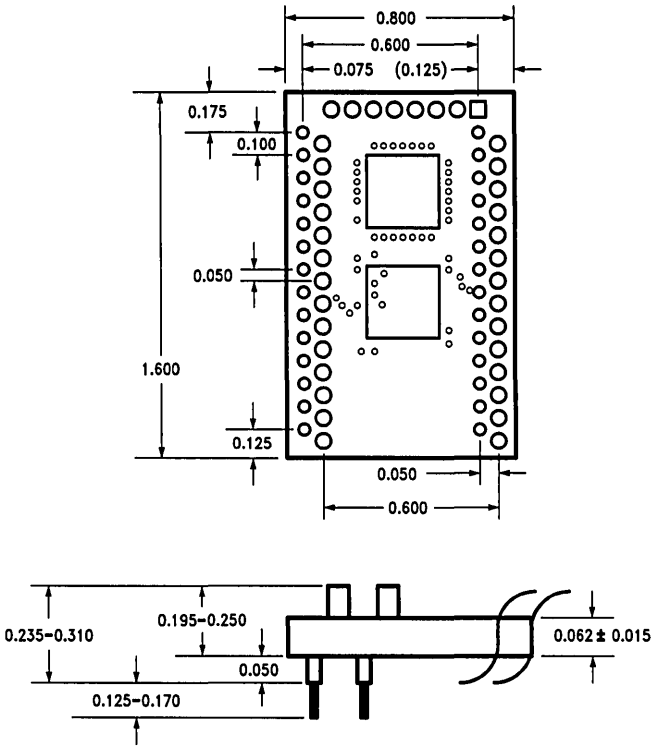


FIGURE 5. Crystal and RC Oscillator Connection Diagrams

TABLE II. Clock Options Per Package

X	Order Part Number	Clock Option
1	COP820CP-1/COP840CP-1	Crystal Oscillator Divide by 10 Option
2	COP820CP-2/COP840CP-2	External Oscillator Divide by 10 Option
3	COP820CP-3/COP840CP-3	RC Oscillator Divide by 10 Option
4	COP820CP-4/COP840CP-4	Crystal Oscillator Divide by 20 Option (High Speed)
5	COP820CP-5/COP840CP-5	External Oscillator Divide by 20 Option

COP820CP/840CP Dimensions Diagram



TL/DD/9683-8

FIGURE 6

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/COP840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. the system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains

DIAL-A-HELPER Users Manual P/N

Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customers problem by actually getting customers system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 baud

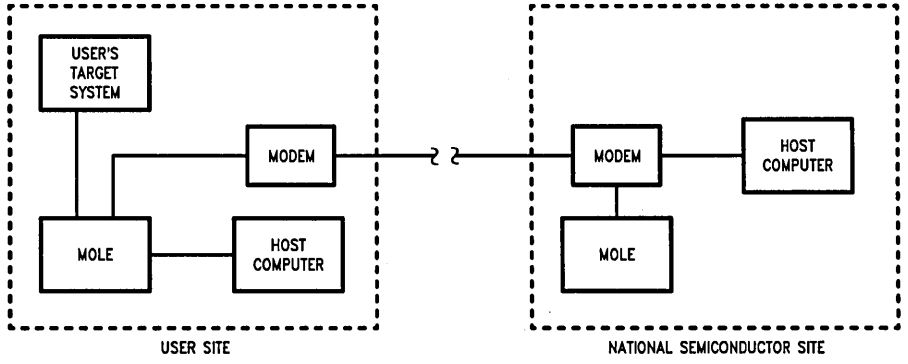
Set-Up: Length: 8-bit

Parity: None

Stop bit: 1

Operation: 24 hrs., 7 days

DIAL-A-HELPER



TL/DD/9683-9

National Semiconductor

PRELIMINARY

COP8720C/COP8721C/COP8722C Single-Chip microCMOS Microcontrollers

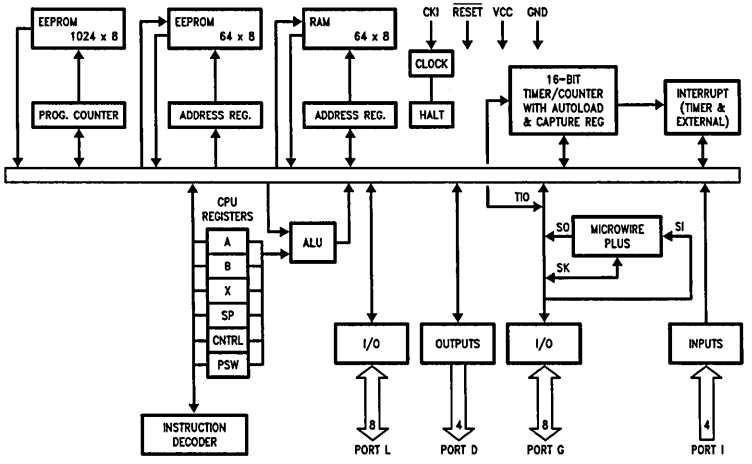
General Description

The COP8720C/COP8721C/COP8722C are members of the COPS™ microcontroller family featuring on-chip EEPROM modules. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP8720C to the specific application. The part operates over a voltage range of 2.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate. The COP8720 is totally compatible with the ROM based COP820C microcontroller. It serves as a form, fit and function emulator device for the COP820 microcontroller family.

Features

- Low Cost 8-bit CORE microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain (2.2 mA at 3 μs instruction rate)
  Extra-low current static HALT mode (Typically < 10 μA)
- Single supply operation: 2.5V to 6.0V
- 1024 bytes EEPROM program memory
- 64 bytes of RAM
- 64 bytes EEPROM data memory
- 16-bit read/write timer operates in a variety of modes
  — Timer with 16-bit auto reload register
  — 16-bit external event counter
  — Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  — Reset master clear
  — External interrupt with selectable edge
  — Timer interrupt or capture interrupt
  — Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instruction
- MICROWIRE/PLUS™ serial I/O
- 28 pin package (optionally 24 or 20 pin package)
- 24 input/output pins
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Form, fit and function EEPROM emulation device for COP820C/COP821C/COP822C
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/9108–1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	50 mA
Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	−65°C to +150°C

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		2.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Operating Voltage during EEPROM Write		4.5		6.0	V
Supply Current (see page 17)					
High Speed Mode, CKI = 20 MHz	V_{CC} = 6V, tc = 1 μs			13	mA
Normal Mode, CKI = 5 MHz	V_{CC} = 6V, tc = 2 μs			7	mA
Normal Mode, CKI = 2 MHz (Note 2)	V_{CC} = 2.5V, tc = 5 μs			2	mA
HALT Current (Note 3)	V_{CC} = 6V, CKI = 0 MHz		<10	30	μA
Input Levels					
RESET, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6.0V, V_{IN} = 0V	−2		+2	μA
Input Pullup Current	V_{CC} = 6.0V, V_{IN} = 0V	40		250	μA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	2.0			mA
All Others					
Source (Weak Pull-Up)	V_{CC} = 4.5V, V_{OH} = 3.2V	10		100	μA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	μA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			mA
TRI-STATE Leakage		−2.0		+2.0	μA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Room Temp) without Latchup (Note 5)				±100	mA
RAM Retention Voltage, Vr	500 ns Rise and Fall Time (Min)		2.0		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports are at TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Human body mode, 100 pF through 1500Ω.

Note 5: Except pins 3, 4, 24
pins 3, 24: +60 mA
pin 4: −25 mA

AC Electrical Characteristics $-40°C < T_A < +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$	1		DC	μs
(Div-by 20)	$2.5V \leq V_{CC} < 4.5V$	2.5		DC	μs
Normal Mode	$V_{CC} \geq 4.5V$	2		DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$	5		DC	μs
R/C Oscillator Mode	$V_{CC} \geq 4.5V$	3		DC	μs
(Div-by 10)					
(See Page 16)	$2.5V \leq V_{CC} < 4.5V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 6)	fr = Max (÷20 Mode)	33		66	%
Rise Time (Note 6)	fr = 20 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 20 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	200			ns
	$2.5V \leq V_{CC} < 4.5V$	500			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	60			ns
	$2.5V \leq V_{CC} < 4.5V$	150			ns
Output Propagation Delay t_{PD1}, t_{PD0}	$R_L = 2.2k, C_L = 100$ pF				
SO, SK	$V_{CC} \geq 4.5V$			0.7	μs
	$2.5V \leq V_{CC} < 4.5V$			1.75	μs
All Others	$V_{CC} \geq 4.5V$			1	μs
	$2.5V \leq V_{CC} < 4.5V$			2.5	μs
MICROWIRE™ Setup Time t_{UWS}		20			ns
MICROWIRE Hold Time t_{UWH}		56			ns
MICROWIRE Output Valid Time t_{UV}				220	ns
Input Pulse Width					
Interrupt Input High Time		t_C			
Interrupt Input Low Time		t_C			
Timer Input High Time		t_C			
Timer Input Low Time		t_C			
Reset Pulse Width		1.0			μs

Note 6: Parameter sampled but not 100% tested.

EEPROM Characteristics

Parameter	Condition	Min	Typ	Max	Units
EEPROM Write Cycle Time	$4.5V \le V_{CC} \le 6.0V$	15	20	25	ms
EEPROM Number of Writes				10000	Cycles

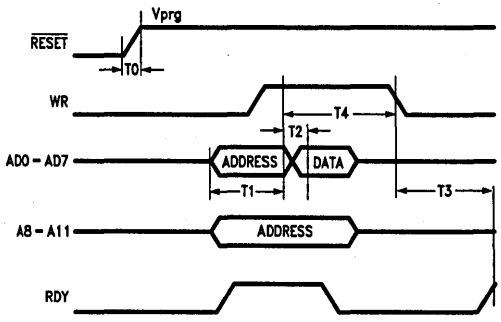
EEPROM DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Symbol	Min	Typ	Max	Units
V_{CC} Level for Write Lock Out	V_{LKO}	3.9		4.4	V
Supply Current	I_{CC}			35	mA
Programming Voltage to $\overline{RESET}$ Pin	V_{prg} $4.5V \le V_{CC} \le 6.0V$	11.5	12	12.5	V
$\overline{RESET}$ Input Current	I_{IH}			2	mA
All Other Inputs, Input Current		−2		2	μA
TRI-STATE Leakage Current		−5		5	μA
Input Low Level	V_{IL}			$0.2\,V_{CC}$	V
Input High Level	V_{IH}	$0.7\,V_{CC}$		$1.0\,V_{CC}$	V
Output Low Level, $I_{OL} = 0.8$ mA	V_{OL}			0.4	V
Output High Level, $I_{OH} = -0.4$ mA	V_{OH}	3.2			V

EEPROM AC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

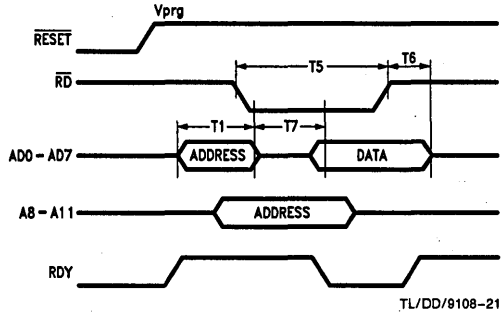
Parameter	Symbol	Min	Typ	Max	Units
CKI Input Frequency	f	10		20	MHz
CKI Duty Cycle		33		66	%
$\overline{RESET}$ Rise Time	T0			1	μs
Address Setup Time	T1			17	t_C
Data Input Valid Time	T2			4	t_C
Program Time	T3	15		25	ms
WR Pulse Width	T4			50	μs
RD Pulse Width	T5			50	t_C
Time to TRI-STATE	T6			17	t_C
Read Access Time	T7			69	t_C

Timing Diagrams



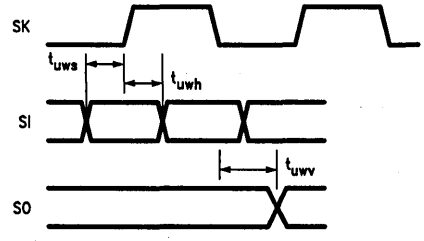
TL/DD/9108-17

FIGURE 2a. COP8720C EEPROM Write Timing Diagrams by Programming Mode



TL/DD/9108-21

FIGURE 2b. COP8720C EEPROM Read Timing Diagrams in Programming Mode

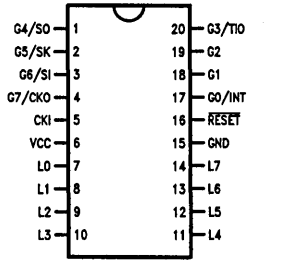


TL/DD/9108-22

FIGURE 2c. MICROWIRE/PLUS Timing Diagram

Connection Diagrams

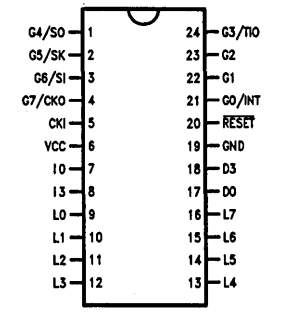
20-Pin Dual-In-Line Package



TL/DD/9108-3

Order Number COP8722CN
See NS Molded Package Number N20A

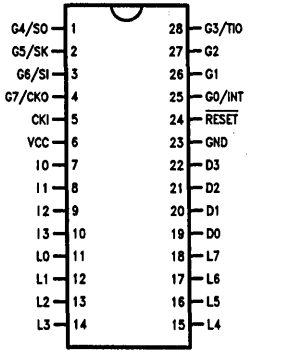
24-Pin Dual-In-Line Package



TL/DD/9108-4

Order Number COP8721CN
See NS Molded Package Number N24A

28-Pin Dual-In-Line Package; PLCC



TL/DD/9108-5

Order Number COP8720CN
See NS Molded Package Number N28B

Order Number COP8720CV
See NS PLCC Package Number V28A

FIGURE 3

Connection Diagrams (Continued)

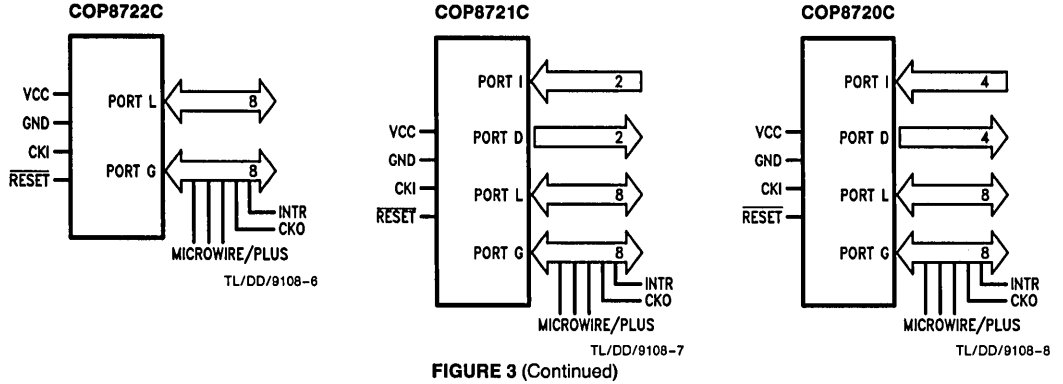


FIGURE 3 (Continued)

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

$\overline{\text{RESET}}$ is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below.

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when $\overline{\text{RESET}}$ goes low.

The D2 pin is sampled at reset. If it is held low at reset the COP8720C enters the ROMless mode of operation.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 15-bit Program Counter register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the program counter stack in RAM during subroutine calls and returns.

Functional Description (Continued)

MEMORY

The COP8720C contains 1 Kbyte of Program EEPROM, 64 bytes of on-chip RAM and Registers, I/O, 64 bytes of Data EEPROM and 256 bytes of firmware ROM.

PROGRAM MEMORY

Program memory for the COP8720C consists of two modules—the 1 Kbyte program EEPROM and the 256 byte ROM which contains the firmware routines for reading and programming the EEPROM.

Memory locations in the 1 Kbyte program EEPROM module are accessed by the address register, EEAR, and the data register, EROMDR. The EEAR is mapped into the address locations E2 and E3. The EROMDR register is located at the address E1.

Under normal conditions, the program EEPROM and the ROM are addressed by the PC and their contents go to the instruction bus. During the EEPROM program and verify cycle, the EEPROM is treated as data memory while the COP8720C is executing out of the firmware ROM. The EEPROM is addressed through the EEAR register. The EROMDR register holds the data read back from the EEPROM location during a verify cycle and holds the data to be written into the EEPROM location during a program cycle. The verify cycle takes 1 instruction cycle and the write cycle takes 20 ms.

Accesses to the program EEPROM is controlled by two flags, AEN and PEN, in the control register, EECR.

AEN	PEN	Access Type
0	0	Normal
0	1	Normal
1	0	EEPROM Read Cycle
1	1	EEPROM Write Cycle

To prevent accidental erasures and over-write situations the application program should not set the AEN and PEN flags in the EECR register. The COP8720C supports application accesses to the EEPROM module via two subroutines in the firmware ROM—an EEPROM read and an EEPROM write subroutine. To program an EEPROM memory location, the user loads the EECR and EROMDR registers and invokes the write subroutine at the address 40C0 Hex. To read an EEPROM location the user loads the EEAR register with the address of the EEPROM memory location and invokes the read subroutine at the address 40D4 Hex. The read subroutine returns the contents of the addressed EEPROM location in the EROMDR register.

DATA MEMORY

The data memory for the COP8720C consists of on-chip RAM, EEPROM, I/O and registers. Data memory is accessed directly by the instruction or indirectly by the B, X and SP registers.

RAM

The COP8720C has 64 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" that can be loaded efficiently, decremented and tested. Three specific registers: B, X and SP are mapped into this space, the other bytes are available for general use.

The instruction set of the COP8720C permits any bit in the data memory to be set, reset or tested. All I/O and the registers (except the A and PC) are memory mapped; therefore, I/O bits and register bits in addition to the normal data RAM can be directly and individually set, reset and tested.

DATA EEPROM

The COP8720C provides 64 bytes of EEPROM for nonvolatile data memory. The data EEPROM can be read and programmed in exactly the same way as the RAM. All instructions that perform read and write operations on the RAM work similarly upon the data EEPROM.

A data EEPROM programming cycle is initiated by an instruction such as X, LD, SBIT or RBIT. The EE memory support circuitry sets the BsyERAM flag in the EECR register immediately upon beginning a data EEPROM write cycle. It will be automatically reset by the hardware at the end of the data EEPROM write cycle. The application program should test the BsyERAM flag before attempting a write operation to the data EEPROM. A second EEPROM write operation while a write operation is in progress will be ignored. The Werr flag in the EECR register is set to indicate the error status.

SIGNATURE AND OPTION REGISTERS

The COP8720C provides a set of six additional registers implemented with EEPROM cells—the Signature and Option registers.

The Signature register is a four-byte register provided for storing ROM code rev. numbers or other application specific information. The Signature register is shadowed behind the data EEPROM cells at addresses 8C to 8F Hex. Two test modes are provided to allow the Signature register to be read or programmed.

The Option register consists of two bytes shadowed behind the addresses 89 and 8B Hex. The Option register allows the COP8720C to be programmed to accurately emulate the different mask options available on the COP820C and the COP8620C.

—	—	—	—	ROMemu	x	0	ERAemu	89 Hex
—	—	—	—	HS	RC	XTAL	x	8B Hex

ROMemu: When set, the Data EEPROM and all the EE related registers become inaccessible. Thus, the EE registers look like nonexistent memory locations when addressed by the application program and the Program EEPROM behaves just like ordinary ROM. Thus, setting the ROMemu bit allows the COP8720C to emulate the ROM based COP820C with 100% accuracy.

ERAemu: When set, the EEAR and the EROMDR become inaccessible. Thus, by setting the ERAemu bit allows the CPO8720C to accurately emulate the COP8620C. Note that the ERAemu is a subset of the ROMemu flag. ROMemu is in effect when both the flags are set.

HS, RC, XTAL: These three bits allow the COP8720C to emulate the clock options of the COP820C. Note that only five out of the possible eight combinations are legal—the combinations 0E, 0C and 06 are illegal combinations.

EECR and EE SUPPORT CIRCUITS

The EEPROM program and data modules share a common set of EE support circuits to generate all necessary high

Functional Description (Continued)

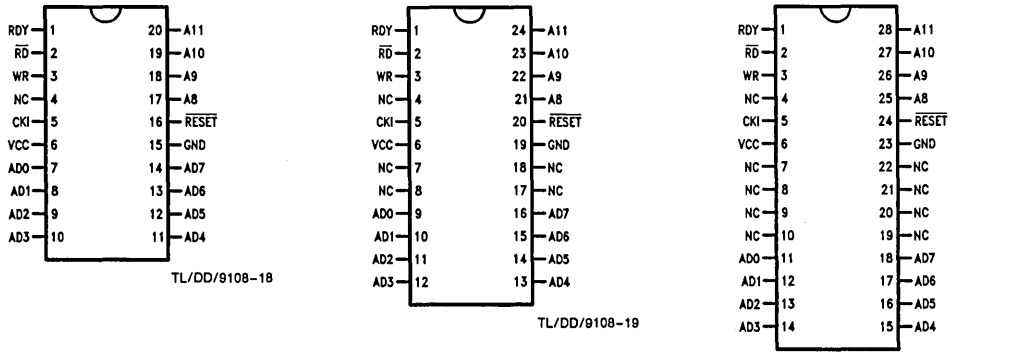


FIGURE 4. Pinouts for the COP8720C in Programming Mode

voltage programming pulses. Each programming cycle consists of a 10 ms erase cycle followed by a 10 ms write cycle for each byte. An EEPROM cell in the erase state is read out as a 0 and the written state is read out as a 1. Since the two EE modules share the support circuitry, programming the two modules at the same time is not allowed.

The EECR register provides control, status and test mode functions for the EE modules.

The EECR register bit assignments are shown below.

EECR Register Bit Assignment

Wr	Test Mode Codes					AEN	PEN	
Rd	Test	Mode	Codes	BsyEROM	BsyERAM	AEN	V_{LKO}	Werr
Bit	7	6	5	4	3	2	1	0

Werr Write Error. Writing to data EEPROM while a previous write cycle is still busy, that is BsyERAM is not 0, causes Werr to be set to 1 indicate error status. Werr is cleared by writing a 0 into it.

PEN A program EEPROM programming cycle is started by setting PEN and AEN to 1 at the same time. PEN is "written thru". It is not latched.

V_{LKO} EECR bit 1 is read as the lock out indicator. A low V_{CC} detector is enabled at the start of the EE programming cycle. If it finds V_{CC} less than V_{LKO}, the V_{LKO} status bit is set and the write cycle is aborted. The V_{LKO} status bit stays latched until the start of another EE programming cycle.

AEN AEN controls the program EEPROM address/data interface. when AEN is 0, the EEPROM is the program memory. It is adressed by PC, and its output data goes onto the instruction bus. When AEN is set to 1, the EEPROM becomes data memory. It is addressed by the EEAR, and it is accessed from the EROMDR.

BsyERAM Set to 1 when data EEPROM is being written, is automatically reset by the hardware upon completion of the write operation.

BsyEROM Set to 1 when program EEPROM is being written, is automatically reset by the hardware upon completion of the write operation.

Bits 3 to 7 of the EECR are used for encoding various EEPROM module test modes, most of which are for factory manufacturing tests. Two of the test modes used for accessing the signature and option registers are described in a previous section. The EE test modes are activated by applying high voltage to the $\overline{\text{RESET}}$ pin. Some of the test modes, if activated improperly, can make the part inoperable. These test modes are reserved for use by the manufacturer only.

The EECR register is cleared by $\overline{\text{RESET}}$. EECR is mapped into address location E0.

When either BsyERAM or BsyEROM is set to 1, that is an EEPROM programming cycle is in progress, the AEN bit is locked up and cannot be changed by the processor.

EXTERNALLY PROGRAMMING THE PROGRAM EEPROM

As shown in the previous section, the COP8720C permits the program EEPROM memory module to be altered under program control via the EECR register. To facilitate ease of development the COP8720C also provides an external mode of loading executable code into the program EEPROM module.

This section describes the programming method for the COP8720C EEPROM.

Programming the COP8720C EEPROM or the special registers is initiated by applying V_{PRG} to the $\overline{\text{RESET}}$ pin. Control gets transferred to the firmware ROM when V_{PRG} is applied to the $\overline{\text{RESET}}$ pin. The program contained in the firmware ROM sets up the I/O of the COP8720C to simulate the I/O requirements of a 2-kbyte memory device. This is done by setting up the COP8720C I/O as eight bits of address/data lines, three address lines, read/write control and a ready signal.

Functional Description (Continued)

Figure 4 shows the three packages and the associated I/O. The pin descriptions are as follows:

V_{CC}	Positive 5V Power Supply
GND	Ground
$\overline{RESET}$	Active Low Reset Input
CKI	Clock Input
AD0–AD7	Multiplexed Address/Data Lines
A8–A11	Address Lines
$\overline{RD}$	Active Low Read Strobe
WR	Active High Write Strobe
RDY	Active High Ready Output

The firmware ROM program allows the user to reference the special registers as EEPROM memory locations in the address range 2048–2070 decimal. The following mapping is used:

Signature Register #1 at EEPROM address 800 Hex
Signature Register #2 at EEPROM address 801 Hex
Signature Register #3 at EEPROM address 802 Hex
Signature Register #4 at EEPROM address 803 Hex

Option Register #1 at EEPROM address 804 Hex
Option Register #2 at EEPROM address 805 Hex

Note that in order to reference these registers the user must come in with addresses in the range 800 Hex to 805 Hex.

PROGRAMMING STEPS

The programmig host has to go through the following steps for the write and verify cycles. (See *Figure 2*)

WRITE:

1. Power is applied with the $\overline{RESET}$ and WR pins low and the $\overline{RD}$ high.
2. $\overline{RESET}$ is then brought up to V_{prg} within 1 μs.
3. The lower byte of the address to be written into is applied to the pins AD0–AD7 and the upper 3 bits of the address applied to the pins A8–A11.
4. Observing the setup times, WR is brought high.
5. The data to be programmed is applied to the pins AD0–AD7.
6. The RDY signal from the COP8720C goes low. This indicates that the WR and data on AD0–AD7 have been accepted and these inputs can be removed.
7. The programming host must now either wait for the RDY signal to go high or wait at least 20 ms before initiating a new programming cycle.

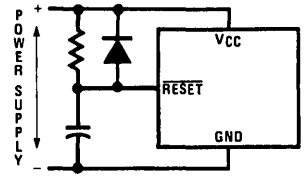
VERIFY:

1. Power is applied with $\overline{RESET}$ and WR pins held low and the $\overline{RD}$ high.
2. The $\overline{RESET}$ pin is brought up to V_{prg} within 1 μs.
3. The lower byte of the address to be read is applied to the pins AD0–AD7 and the upper three bits to the pins AD8–AD11.
4. Observing setup times the $\overline{RD}$ pin is brought low.
5. After a time T7, the RDY signal from the COP8720C goes low and data is ready for the host on the pins AD0–AD7. The data stays until the $\overline{RD}$ signal goes back high after which the RDY signal will go back high.
6. The host must wait for the RDY signal to go back high before the next read cycle is initiated.

RESET

The $\overline{RESET}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{RESET}$ input is pulled low. Upon initialization, the ports L and G are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L & G are cleared.

The external RC network shown in *Figure 5* should be used to ensure that the $\overline{RESET}$ pin is held low until the power supply to the chip stabilizes. It is recommended that the components of the RC network be selected to provide a $\overline{RESET}$ delay of at least five times the power supply rise time or the minimum $\overline{RESET}$ pulse width, whichever is greater.



TL/DD/9108–9

FIGURE 5. Recommended Reset Circuit

OSCILLATOR CIRCUITS

Figure 6 shows the three clock oscillator configurations available for the COP8720C.

A. CRYSTAL OSCILLATOR

The COP8720C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

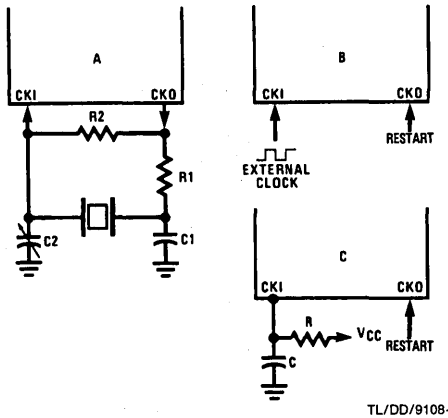
B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9108–10

FIGURE 6. Crystal and R-C Connection Diagrams

OSCILLATOR OPTIONS

The COP8720C can be driven by clock inputs between DC and 20 MHz. For low input clock frequencies ($\leq$ 5 MHz) the instruction cycle frequency can be selected to be the input clock frequency divided by 10. This mode is known as the Normal Mode.

Functional Description (Continued)

TABLE I. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	20	V_{CC} = 5V
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4	V_{CC} = 2.5V
0	1	200	100–150	0.455	V_{CC} = 2.5V

TABLE II. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8–2.2	3 to 6	V_{CC} = 5V
5.6	100	1.5–1.1	6 to 11	V_{CC} = 5V
6.8	100	1.1–0.8	7.5 to 18	V_{CC} = 2.5V

For oscillator frequencies that are greater than 5 MHz the chip must run with a divide by 20. This is known as the High Speed mode.

The COP820C microcontroller has five mask options for configuring the clock input. To emulate these mask options 3 bits must be set in the Option register.

HS	RC	XTAL	Mask Option
1	0	1	High Speed Crystal
0	0	1	Normal Mode Crystal
1	0	0	High Speed External
0	0	0	Normal Mode External
0	1	0	R/C Oscillator

The CKI and CKO pins are automatically configured upon selecting a particular option.

— High Speed Crystal (CKI/20) CKO for crystal configuration

— Normal Mode Crystal (CKI/10) CKO for crystal configuration

— High Speed External (CKI/20) CKO available as G7 input

— Normal Mode External (CKI/10) CKO available as G7 input

— R/C (CKI/10) CKO available as G7 input

Where, G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

1) Oscillator operating mode—I1

2) Internal switching current—I2

3) Internal leakage current—I3

4) Output source current—I4

5) DC current caused by external input not at V_{CC} or GND—I5

Thus the total current drain, It is given as

$$It = I1 + I2 + I3 + I4 + I5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$I2 = C \times V \times f$

Where

C = equivalent capacitance of the chip. (TBD)

V = operating voltage

f = CKI frequency

The typical capacitance for the COP820C is TBD pF.

Some sample current drain values at V_{CC} = 6V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
20	1	13
3.58	3	2.2
2	5	1.2
0.3	33	0.2
0 (HALT)	—	<0.01

HALT MODE

The COP8720C supports a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to Vr (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the $\overline{\text{RESET}}$ or by the CKO pin. A low on the $\overline{\text{RESET}}$ line reinitializes the

Functional Description (Continued)

microcontroller and start executing from the address 0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction.

INTERRUPTS

The COP8720C has a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control).

A maskable interrupt on timer carry or timer capture.

A non-maskable software/error interrupt on opcode zero.

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and continues from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Either of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

DETECTION OF ILLEGAL CONDITIONS

The COP8720C incorporates a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP8720C grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP8720C to interface with any of National Semiconductor's Microwire peripherals (i.e. A/D converters, display drivers, etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 8* shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.

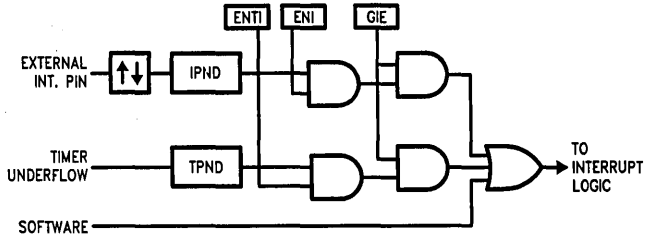
The CNTRL register is used to configure and control the MICROWIRE mode. To use the MICROWIRE, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	$2t_C$
0	1	$4t_C$
1	x	$8t_C$

where,

t_C is the instruction cycle clock.



TL/DD/9108–11

FIGURE 7. Interrupt Block Diagram

Functional Description (Continued)

MICROWIRE PLUS OPERATION

Setting the BUSY bit in the PSW register causes the Microwire arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP8720C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 9* shows how two COP8720C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

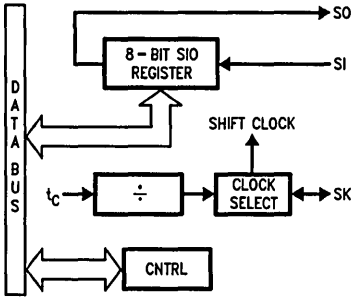
Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP8720C. The MICROWIRE Master always initiates all data exchanges. (See *Figure 9.*) The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summaries the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See *Figure 9.*)



TL/DD/9108–12

FIGURE 8. MICROWIRE Block Diagram

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The COP8720C has a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes.

MODE 1. TIMER WITH AUTO-LOAD REGISTER

In this mode of operation the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allow the generation of square-wave outputs or pulse width modulated outputs under software control. (See *Figure 10.*)

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See *Figure 10.*)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See *Figure 11.*)

Functional Description (Continued)

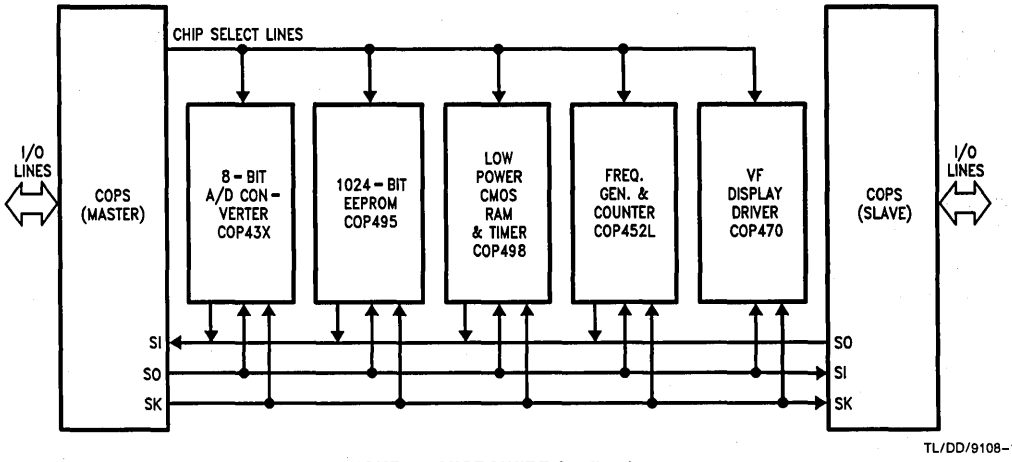


FIGURE 9. MICROWIRE Application

TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t_C
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t_C
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t_C
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t_C

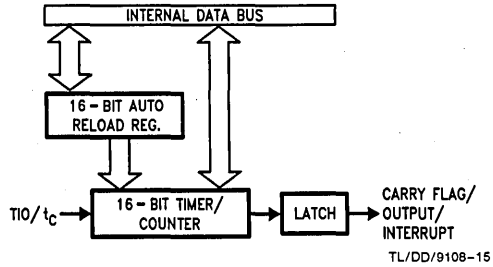


FIGURE 10. Timer/Counter Auto Reload Mode Block Diagram

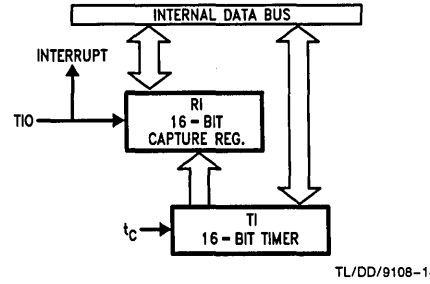
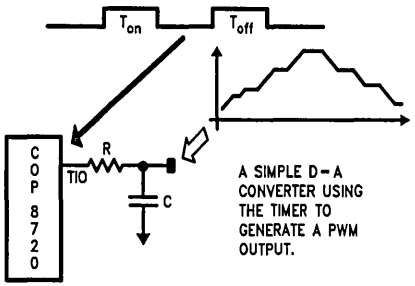


FIGURE 11. Timer Capture Mode Block Diagram

Functional Description (Continued)

TIMER PWM APPLICATION

Figure 12 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.



TL/DD/9108-16

FIGURE 12. Timer Application

Control Registers

CNTRL REGISTER (ADDRESS X'OOEE)

The Timer and MICROWIRE control register contains the following bits:

S1 & S0 Select the MICROWIRE clock divide-by

IEDG External interrupt edge polarity select (0 = rising edge, 1 = falling edge)

MSEL Enable MICROWIRE functions S0 and SK

TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)

TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)

TC2 Selects the capture mode

TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0

BIT 7 BIT 0

PSW REGISTER (ADDRESS x'00EF)

The PSW register contains the following select bits:

GIE Global interrupt enable

ENI External interrupt enable

BUSY MICROWIRE busy shifting

IPND External interrupt pending

ENTI Timer interrupt enable

TPND Timer interrupt pending

C Carry Flag

HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE

Bit 7 Bit 0

Operating Modes

These controllers have two operating modes: Single Chip mode and the ROMless mode. The operating mode is determined by the state of the D2 pin at power on reset.

SINGLE CHIP MODE

In the Single Chip mode, the controller functions as a self contained microcontroller. It can address internal RAM and ROM. All ports configured as memory mapped I/O ports.

ROMLESS MODE

The COP8720C will enter the ROMless mode of operation if the D2 pin is held at logical "0" at reset. In this case the internal PROGRAM EEPROM is disabled and the controller can now address up to 32 kbytes of external program memory. It continues to use the on board RAM, and DATA EEPROM.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
00 to 2F	On Chip RAM Bytes
30 to 7F	Unused RAM Address Space (Reads as all Ones)
80 to BF	64 Bytes DATA EEPROM
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8-DB	Reserved for Port C
DC-DF	Port D
E0 to EF	On Chip Functions and Registers
E0	EECR
E1	EROMDR
E2	EEAR Low Byte
E3	EEAR High Byte
E4-E8	Reserved
E9	MICROWIRE Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Memory Map (Continued)

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for the COP8720C. The operand is the memory addressed by the B register or X register.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from −31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
MemI	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

Instruction Set

ADD	add	A ← A + Meml
ADC	add with carry	A ← A + Meml + C, C ← Carry HC ← Half Carry
SUBC	subtract with carry	A ← A + $\overline{\text{Meml}}$ + C, C ← Carry HC ← Half Carry
AND	Logical AND	A ← A and Meml
OR	Logical OR	A ← A or Meml
XOR	Logical Exclusive-OR	A ← A xor Meml
IFEQ	IF equal	Compare A and Meml, Do next if A = Meml
IFGT	IF greater than	Compare A and Meml, Do next if A > Meml
IFBNE	IF B not equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Decrement Reg. ,skip if zero	Reg ← Reg − 1, skip if Reg goes to 0
SBIT	Set bit	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	Reset bit	0 to bit, Mem
IFBIT	If bit	If bit, Mem is true, do next instr.
X	Exchange A with memory	A ⟷ Mem
LD A	Load A with memory	A ← Meml
LD mem	Load Direct memory Immed.	Mem ← Imm
LD Reg	Load Register memory Immed.	Reg ← Imm
X	Exchange A with memory [B]	A ⟷ [B] (B ← B ± 1)
X	Exchange A with memory [X]	A ⟷ [X] (X ← X ± 1)
LD A	Load A with memory [B]	A ← [B] (B ← B ± 1)
LD A	Load A with memory [X]	A ← [X] (X ← X ± 1)
LD M	Load Memory Immediate	[B] ← Imm (B ← B ± 1)
CLRA	Clear A	A ← 0
INCA	Increment A	A ← A + 1
DECA	Decrement A	A ← A − 1
LAID	Load A indirect from ROM	A ← ROM(PU,A)
DCORA	DECIMAL CORRECT A	A ← BCD correction (follows ADC, SUBC)
RRCA	ROTATE A RIGHT THRU C	C → A7 → . . . → A0 → C
SWAPA	Swap nibbles of A	A7 . . . A4 ⟷ A3 . . . A0
SC	Set C	C ← 1, HC ← 1
RC	Reset C	C ← 0, HC ← 0
IFC	If C	If C is true, do next instruction
IFNC	If not C	If C is not true, do next instruction
JMPL	Jump absolute long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Jump absolute	PC11..0 ← i (i = 12 bits)
JP	Jump relative short	PC ← PC + r (r is −31 to +32, not 1)
JSRL	Jump subroutine long	[SP] ← PL,[SP-1] ← PU,SP-2,PC ← ii
JSR	Jump subroutine	[SP] ← PL,[SP-1] ← PU,SP-2,PC11..0 ← i
JID	Jump indirect	PL ← ROM(PU,A)
RET	Return from subroutine	SP+2,PL ← [SP],PU ← [SP-1]
RETSK	Return and Skip	SP+2,PL ← [SP],PU ← [SP-1],Skip next instruction
RETI	Return from Interrupt	SP+2,PL ← [SP],PU ← [SP-1],GIE ← 1
INTR	Generate an interrupt	[SP] ← PL,[SP−1] ← PU,SP-2,PC ← 0FF
NOP	No operation	PC ← PC + 1

OPCODE LIST

Bits 7–4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
JP -15	JP -31	LD 0F0, #i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0,[B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1, #i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A,[B]	IFBIT 1,[B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2, #i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A,[B]	IFBIT 2,[B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3, #i	DRSZ 0F3	X A, [X−]	X A, [B−]	IFGT A, #i	IFGT A,[B]	IFBIT 3,[B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4, #i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5, #i	DRSZ 0F5	*	JID	AND A, #i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6, #i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A,[B]	IFBIT 6,[B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7, #i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	IFBIT 7,[B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8, #i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9, #i	DRSZ 0F9	*	*	*	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA, #i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+], #i	INCA	SBIT 2,[B]	RBIT 2,[B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB, #i	DRSZ 0FB	LD A, [X−]	LD A, [B−]	LD [B−], #i	DECA	SBIT 3,[B]	RBIT 3,[B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC, #i	DRSZ 0FC	LD Md, #i	JMPL	X A,Md	*	SBIT 4,[B]	RBIT 4,[B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP +13	C
JP -2	JP -18	LD 0FD, #i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5,[B]	RBIT 5,[B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP +14	D
JP -1	JP -17	LD 0FE, #i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00−0EFF	JMP 0E00−0EFF	JP + 31	JP +15	E
JP -0	JP -16	LD 0FF, #1	DRSZ 0FF	*	*	*	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

Bits 3–0

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time (1 μs at 20 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 20 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B,Imm				1/1			(If B < 16)
LD B,Imm				2/3			(If B > 15)
LD Mem,Imm	2/2		3/3		2/2		
LD Reg,Imm				2/3			

* = > Memory location addressed by B or X or directly.

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Bytes and Cyles per Instruction (Continued)

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs, and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

To program the COP8720C, a special adapter board is provided. This adapter board contains a socket for the COP8720C and plugs directly into the MOLE prom programmer.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/ COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package Contains:
Dial-A-Helper User's Manual Pin
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

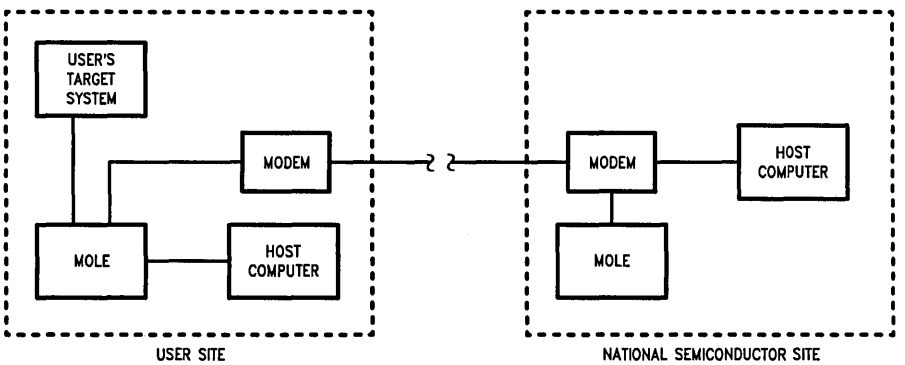
Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting the customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Voice:	(408) 721-5582
Modem:	(408) 739-1162
Baud:	300 or 1200 Baud
Setup:	Length: 8-Bit
	Parity: None
	Stop Bit: 1
Operation:	24 Hours, 7 Days

Dial-A-Helper



TL/DD/9108-23

National Semiconductor

ADVANCE INFORMATION

COP888CL
Single-Chip microCMOS Microcontroller

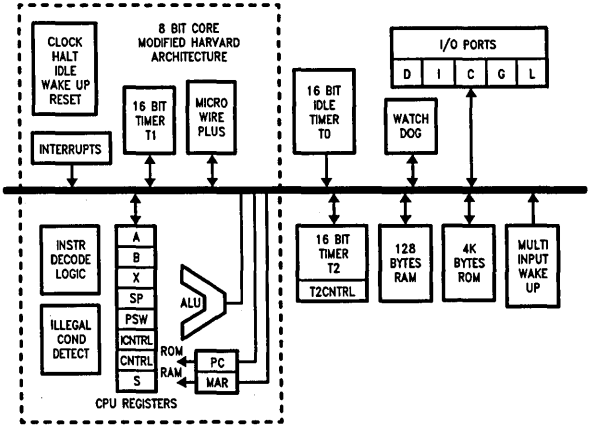
General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. The COP888CL is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 4096 bytes on-board ROM
- 128 bytes on-board RAM
- Single supply operation: 2.5V–6V
- MICROWIRE/PLUS™ serial I/O
- Watch Dog and Clock Monitor logic
- Idle Timer
- Two 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Timers TA, TB (Each with 2 Interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC or 40 N or 28 N or 28 PCC
  - — 44 PCC with 39 I/O pins
  - — 40 N with 36 I/O pins
  - — 28 PCC or 28 N, each with 23 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Extended temperature range: −55°C to +125°C
- ROMless mode for accurate emulation and external program capability
- Single chip COP8XX piggy back emulation device
- Real time emulation and full program debug offered by National's MOLE™ Development System

Block Diagram



TL/DD/9766–1

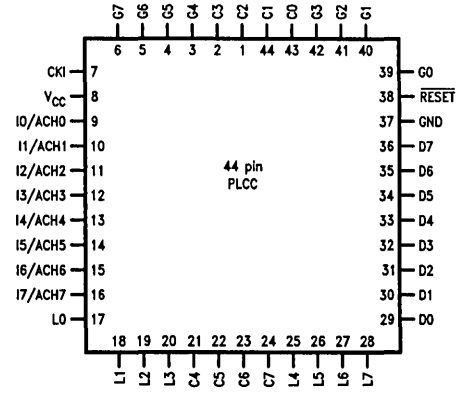
FIGURE 1. COP888CL Block Diagram

General Description (Continued)

It is a fully static part, fabricated using double-metal-silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CL operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The COP888CL may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Connection Diagrams

Plastic Chip Carrier

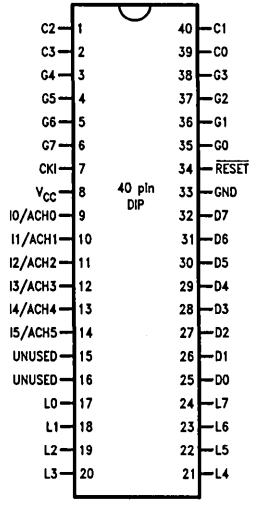


TL/DD/9766-2

Top View

Order Number COP888CL-XXX/V
See NS Plastic Chip Package Number V44A

Dual-In-Line Package

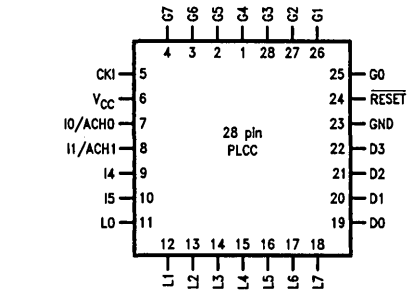


TL/DD/9766-4

Top View

Order Number COP888CL-XXX/N
See NS Molded Package Number N40A

Plastic Chip Carrier



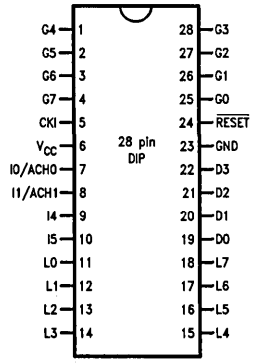
TL/DD/9766-3

Top View

Order Number COP884CL-XXX/V
See NS Plastic Chip Package Number V28A

***Note:** Unused pins must be connected to GND.

Dual-In-Line Package



TL/DD/9766-5

Top View

Order Number COP884CL-XXX/N
See NS Molded Package Number N28A

FIGURE 2. COP888CL Connection Diagrams

Connection Diagrams (Continued)

COP888CL Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU		12	18	18
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	CKO			4	6	6
D0	O	ROM DATA+		19	25	29
D1	O	PCL+		20	26	30
D2	O	EMUL+		21	27	31
D3	O	PCU+		22	28	32
I0	I	ACH0		7	9	9
I1	I	ACH1		8	10	10
I2	I	ACH2			11	11
I3	I	ACH3			12	12
I4	I	ACH4		9	13	13
I5	I	ACH5		10	14	14
I6	I	ACH6				15
I7	I	ACH7				16
D4	O	S CLOCK+			29	33
D5	O	HALTSEL+			30	34
D6	O	LOAD+			31	35
D7	O	D DATA+			32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
Unused					16	
Unused					15	
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

– = Unbonded Pins
+ = Only in the ROMless Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage				6	V
Power Supply Ripple (Note 1)	Peak-to-Peak	2.5		0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	V_{CC} = 6V, t_c = 1 μs			15	mA
CKI = 4 MHz	V_{CC} = 2.5V, t_c = 2.5 μs			2	mA
HALT Current (Note 3)	V_{CC} = 6V, CKI = 0 MHz		<1		μA
IDLE Current					
CKI = 10 MHz	V_{CC} = 6V, t_c = 1 μs			5	mA
CKI = 4 MHz	V_{CC} = 2.5V, t_c = 2.5 μs			0.6	mA
Input Levels					
Reset					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6V, V_{IN} = 0V	−2		+2	μA
Input Pullup Current	V_{CC} = 6V, V_{IN} = 0V	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4V, V_{OH} = 3.3V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4V, V_{OL} = 1V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.2			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4V, V_{OH} = 2.7V	10		100	μA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	μA
Source (Push-Pull Mode)	V_{CC} = 4V, V_{OH} = 3.3V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink (Push-Pull Mode)	V_{CC} = 4V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			mA
TRI-STATE Leakage		−2		+2	μA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified (Continued)

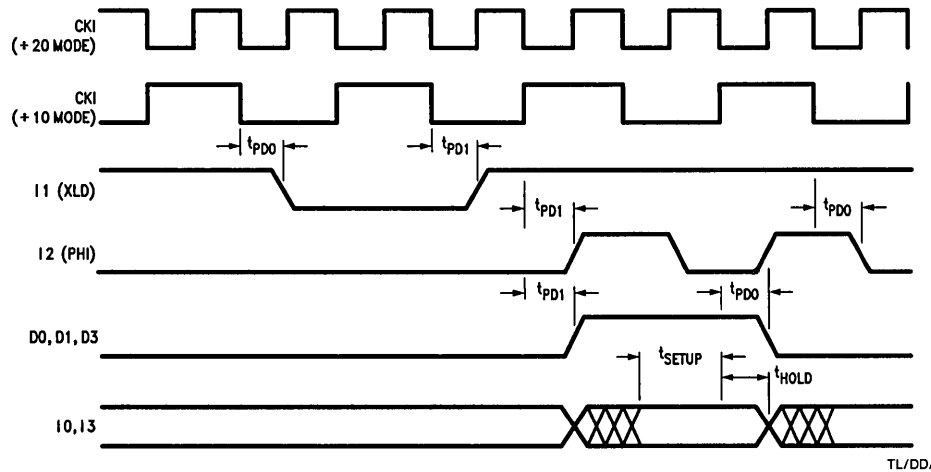
Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup			200		mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)		2		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator, or	$4V \le V_{CC} \le 6V$	1		DC	μs
External Oscillator	$2.5V \le V_{CC} < 4V$	2.5		DC	μs
R/C Oscillator	$4V \le V_{CC} \le 6V$	3		DC	μs
	$2.5V \le V_{CC} < 4V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 5)	f_r = Max	40		60	%
Rise Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4V \le V_{CC} \le 6V$	200			ns
	$2.5V \le V_{CC} < 4V$	500			ns
t_{HOLD}	$4V \le V_{CC} \le 6V$	60			ns
	$2.5V \le V_{CC} < 4V$	150			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4V \le V_{CC} \le 6V$			0.7	μs
	$2.5V \le V_{CC} < 4V$				
All Others	$4V \le V_{CC} \le 6V$			1	μs
	$2.5V \le V_{CC} < 4V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UV})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

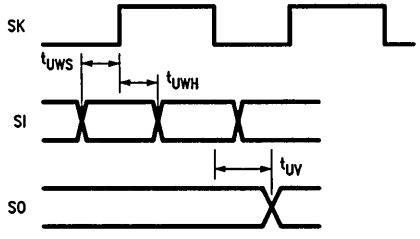
Note 5: Parameter sample but not 100% tested.

AC Electrical Characteristics (Continued)



TL/DD/9766-25

FIGURE 2a. AC Timing Diagrams in ROMless Mode



TL/DD/9766-26

FIGURE 2b. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CL contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CL memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CL. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Pin Descriptions (Continued)

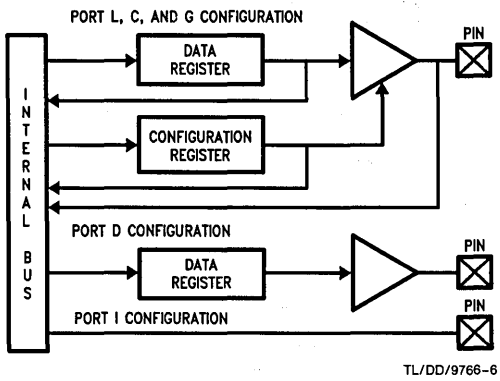


FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wakeup (MIWU) on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B.

Port L has the following alternate features:

- L0 MIWU
- L1 MIWU
- L2 MIWU
- L3 MIWU
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 serves as the dedicated CKO clock output. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin or general purpose input (R/C clock configuration), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLK Delay	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE™ Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT WatchDog and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

Port I is an 8-bit Hi-Z input port. The 40-pin device does not have a full complement of Port I pins. The unused pins are not terminated and must be tied to GND.

The 28-pin device has four I pins (I0, I1, I4, I5). The user should pay attention when reading port I to the fact that I4 and I5 are in bit positions 4 and 5 rather than 2 and 3.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CL is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CL architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_C) cycle time.

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)
PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with RESET.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CL consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CL vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE counter). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The COP888CL has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

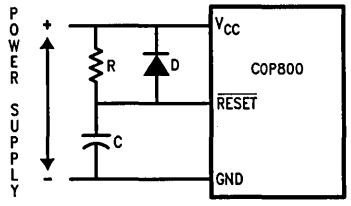
The instruction set of the COP888CL permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CL (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for Ports L, G, and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is initialized high with RESET. The PC, PSW, CNTRL, ICNTRL, and T2CNTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The Stack Pointer, SP, is initialized to 06F Hex.

The COP888CL comes out of RESET with both the WatchDog logic and the Clock Monitor detector armed, and with both the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor detector circuits are inhibited during RESET. The WatchDog service window bits are initialized to the maximum WatchDog service window of 64k t_C clock cycles. The Clock Monitor bit is initialized high, and will cause a Clock Monitor error following RESET if the clock has not reached the minimum specified frequency at the termination of RESET. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16–32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 4* should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes. It is recommended that the components of the RC network be selected to provide a RESET delay of at least five times the power supply rise time or the minimum RESET pulse width, whichever is greater.



TL/DD/9766–7

RC > 5 × Power Supply Rise Time

FIGURE 4. Recommended RESET Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

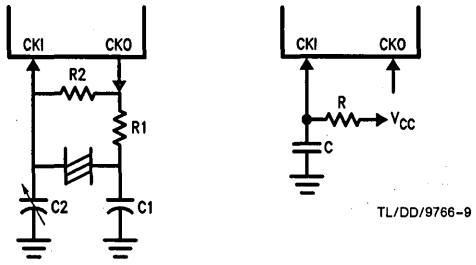


FIGURE 5. Crystal and R/C Oscillator Diagrams

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5

Thus the total current drain, It, is given as

$$It = I1 + I2 + I3 + I4 + I5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Some sample current drain values at V_{CC} = 5V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)		0.005

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE control register contains the following bits:

SL1 & SL0	Select the MICROWIRE clock divide by (00 = 2, 01 = 4, 1× = 8)
IEDG	External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
MSEL	Selects G5 and G4 as MICROWIRE signals SK and SO respectively
T1C0	Timer T1 Start/Stop control in timer modes 1 and 2 Timer T1 Underflow Interrupt Pending Flag in timer mode 3
T1C1	Timer T1 mode control bit
T1C2	Timer T1 mode control bit
T1C3	Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0

Bit 7 Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt
BUSY	MICROWIRE busy shifting flag
EXPND	External interrupt pending
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
T1PNDA	Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
C	Carry Flag
HC	Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE

Bit 7 Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

Control Registers (Continued)

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

- T1ENB Timer T1 Interrupt Enable for T1B Input capture edge
- T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge
- μWEN Enable MICROWIRE/PLUS interrupt
- μWPND MICROWIRE/PLUS interrupt pending
- T0EN Timer T0 Interrupt Enable (Bit 12 toggle)
- T0PND Timer T0 Interrupt pending
- LPEN L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

  Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PNDB	T1ENB
Bit 7							Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

- T2ENB Timer T2 Interrupt Enable for T2B Input capture edge
- T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge
- T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
- T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
- T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3
- T2C1 Timer T2 mode control bit
- T2C2 Timer T2 mode control bit
- T2C3 Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

Emulation and ROMless Modes

The COP888CL can address up to 32 kbytes of address space. If at power up, D2 is held at ground, the COP888CL executes from external memory. Port D is used to interface to external program memory. The address comes out in a serial fashion and the data from the external program memory is read back in a serial fashion. The Port D pins perform the following functions.

- D0 Shifts in ROM data
- D1 Shifts out lower eight bits of PC
- D2 Places the μC in the ROMless mode if grounded at reset
- D3 Shifts out upper eight bits of PC
- D4 Data Shift Clock
- D5 HALT Mask Option select pin (D5 = 0) for HALT enable, D5 = 1 for HALT disable)
- D6 Load Clock
- D7 Shifts out recreated Port D data

The most significant bit of the data to come out on the D3 pin is a status signal.. It is used by the MOLE development system. This "lost" output port (D0–D7) can be accurately reconstructed with external components as shown in *Figure 6*, providing an accurate emulation.

The 44-pin and 40-pin versions of the COP888CL have a full complement of the D Port pins and can be used in the ROMless mode.

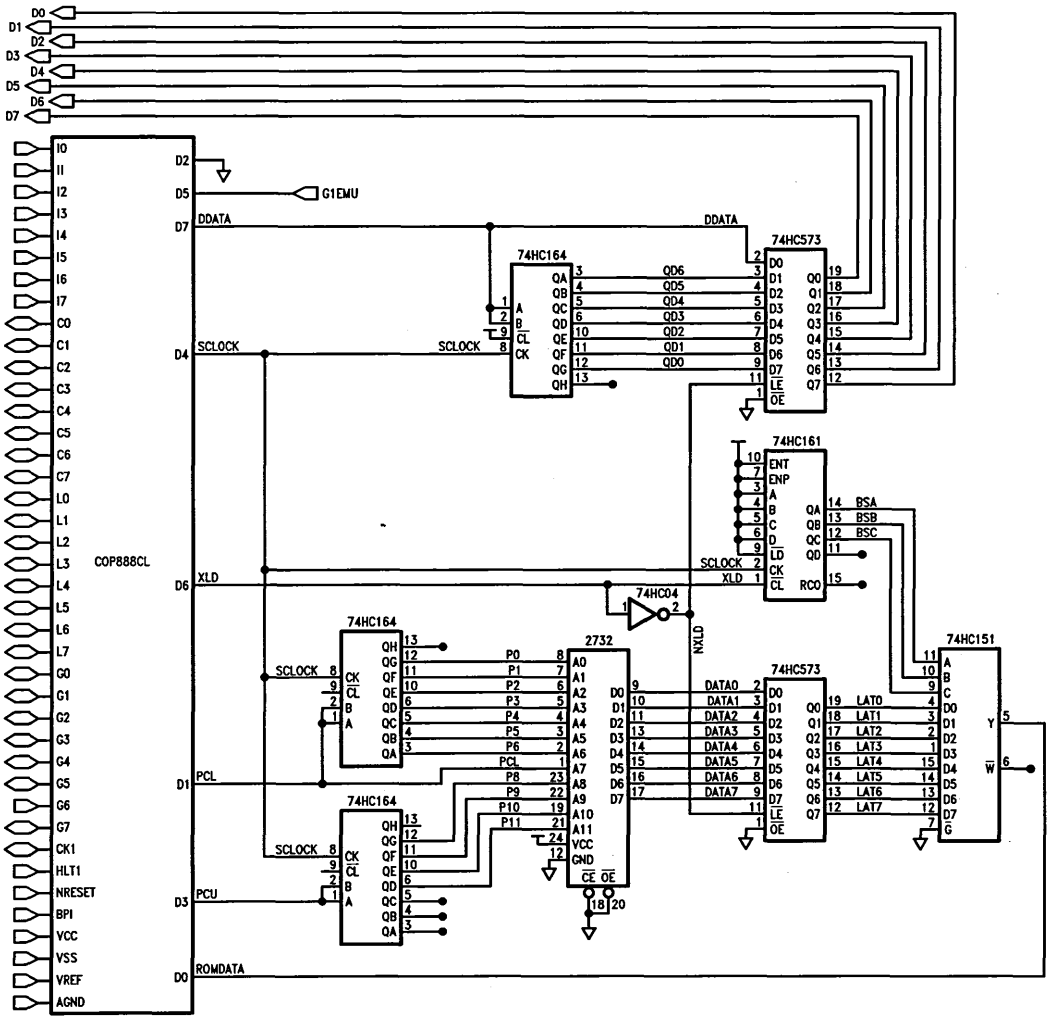
The 28-pin part cannot be used for emulation since it does not have the full complement of 8 D Port pins necessary for entering the ROMless mode.

Note that in the ROMless mode the D Port is recreated one full clock cycle behind the normal port timings.

Note: Standard parts used in the ROMless mode will operate only at a reduced frequency (to be defined).

The COP888CL device has a spare D pin (D5) in the emulation mode since only seven pins are required for emulation and recreation. This pin D5 is used in the emulation mode to enable or disable the HALT mask option feature.

Figure 6 shows the COP888CL Emulation Mode Schematic.



TL/DD/9766–10

FIGURE 6. COP888CL Emulation Mode Schematic

Power Save Modes

The COP888CL offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CL is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock, timers, are stopped. The WatchDog logic on the COP888CL is disabled during the HALT mode. However, the clock monitor circuitry remains active. In the HALT mode, the power requirements of the COP888CL are minimal and the applied voltage (V_{CC}) may be decreased to V_r (V_r = 2.0V) without altering the state of the machine.

The COP888CL supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method preludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET input low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE time is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared at reset.

The COP88CL has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CL will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CL cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit remains active during HALT mode in order to ensure a clock monitor error if the COP888CL inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CL is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the WatchDog logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

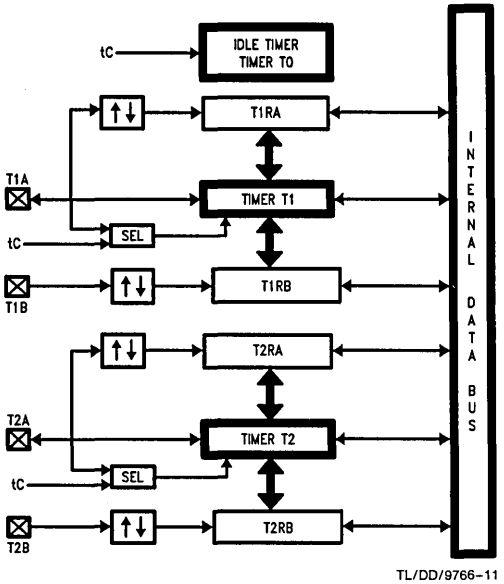
As with the HALT mode, the COP888CL can be returned to normal operation with a RESET, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the twelfth bit (representing 4.096 ms at internal clock frequency of 1 MHz (t_c = 1 μs)) of the IDLE Timer toggles.

This toggle condition of the twelfth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

The user has the option of being interrupted with a transition on the twelfth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the COP888CL will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CL will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.



TL/DD/9766–11

FIGURE 7. Timers for the COP888CL

Timers

The COP888CL contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

Figure 7 shows a block diagram for the timers on the COP888CL.

TIMER T0 (IDLE TIMER)

The COP888CL supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_C. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)
WatchDog logic (See WatchDog description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the twelfth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency (t_C = 1 μs). A control flag T0EN allows the interrupt from the twelfth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1 AND TIMER T2

The COP888CL has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2, are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CL to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CL to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode.

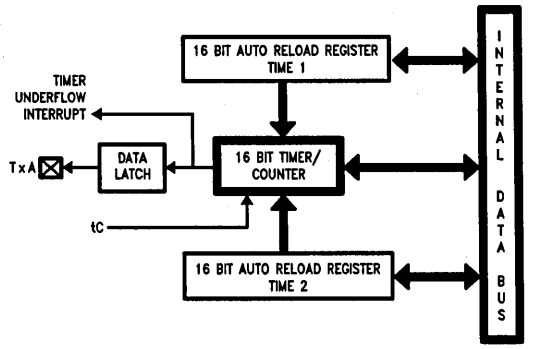
The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.



TL/DD/9766-13

FIGURE 8. Timer in PWM Mode

Timers (Continued)

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

Mode 3. Input Capture Mode

The COP888CL can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_C rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

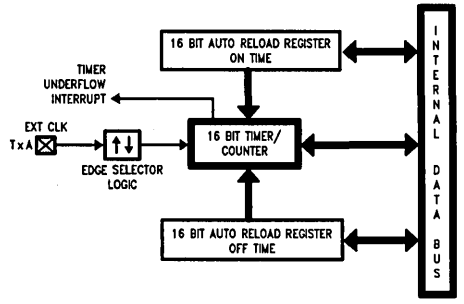
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

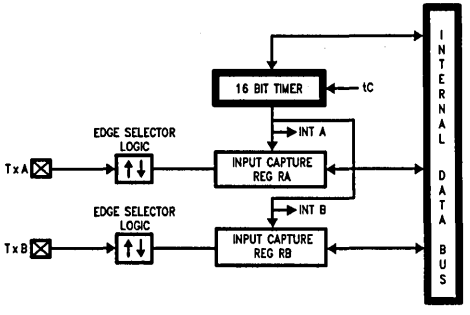
The timers T1 and T2 have indentical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control



TL/DD/9766-14

FIGURE 9. Timer in External Event Counter Mode



TL/DD/9766-15

FIGURE 10. Timer in Input Capture Mode

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_C
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_C
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C

Detection of Illegal Conditions

The COP888CL will detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during RESET. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F Hex is read as all 1's, which in turn will cause the program to return to address FFFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

a. Executing from undefined ROM
b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following RESET, but might not contain the same program initialization procedures).

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CL from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Multi-Input Wakeup (Continued)

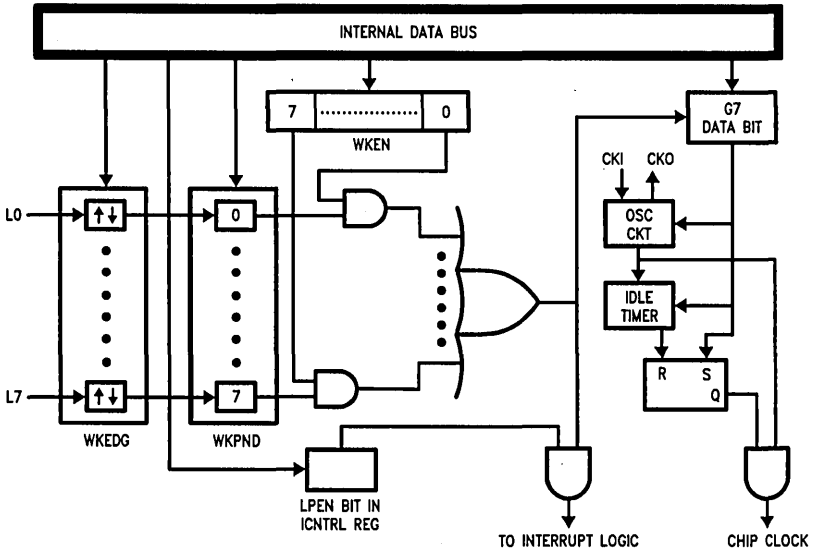


FIGURE 11. Multi-Input Wake Up Logic

Figure 11 shows the Multi-Input Wakeup logic for the COP888CL microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CL to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following RESET, since the L port inputs are left floating as a result of RESET.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called Reg: WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since the Reg: WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CL will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

All three registers Reg:WKEN, Reg:WKPND and Reg:WKEDG are read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG

Multi-Input Wakeup (Continued)

specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CL out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CL will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CL will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CL to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_C instruction cycle clock. The t_C clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during RESET, so the clock start up delay is not present following RESET with the RC clock options.

Interrupts

The COP888CL supports a vectored interrupt scheme. It supports a total of ten interrupt sources. The following table lists all the possible COP888CL interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	T0 Bit 12 Toggle	0yF8–0yF9
(4)	Timer T1	T1 Underflow/ T1A Capture Edge	0yF6–0yF7
(5)	Timer T1	T1B Capture Edge	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
	Reserved	for UART	0yEE–0yEF
	Reserved	for UART	0yEC–0yED
(7)	Timer T2	T2 Underflow/ T2A Capture Edge	0yEA–0yEB
(8)	Timer T2	T2B Capture Edge	0yE8–0yE9
	Reserved	for Future Use	0yE6–0yE7
	Reserved	for Future Use	0yE4–0yE5
(9)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(10) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page.

Interrupts (Continued)

instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_C cycles to execute.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 12 shows the COP888CL Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to RESET, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit **(not accessible by the user)** is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction.

It is cleared by RESET and by the RPND instruction.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

The COP888CL contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

WatchDog

The COP888CL WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDCNT which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table I shows the WDCNT register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 ad 6 of the WDCNT register allow the user to pick an upper limit of the service window.

Table II shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDCNT register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDCNT Register is the Clock Monitor Select bit.

TABLE I

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

WatchDog (Continued)

TABLE II

WDCNT Bit 7	WDCNT Bit 6	Service Window (Lower-Upper Limits)
0	0	2k-8k t_c Cycles
0	1	2k-16k t_c Cycles
1	0	2k-32k t_c Cycles
1	1	2k-64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the COP888CL can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WatchDog Operation

The WatchDog and Clock Monitor are disabled during $\overline{\text{RESET}}$. The COP888CL comes out of $\overline{\text{RESET}}$ with the WatchDog armed, the WatchDog Window Select bits (bits 6, 7 of the WDCNT Register) set, and the Clock Monitor bit (bit 0 of the WDCNT Register) enabled. Thus, a Clock Monitor error will occur after coming out of $\overline{\text{RESET}}$, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDCNT register can be written to only once after $\overline{\text{RESET}}$ and the key data (bits 5 through 1 of the WDCNT Register) must match to be a valid write. This write to the WDCNT register involves two irrevocable choices: (i) the selection of the WatchDog service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDCNT Register involves selecting or deselecting the Clock Monitor, select the WatchDog service window and match the WatchDog key data. Subsequent writes to the WDCNT register will compare the value being written by the user to the WatchDog service window value and the key data (bits 7 through 1) in the WDCNT Register. Table III shows the sequence of events that can occur.

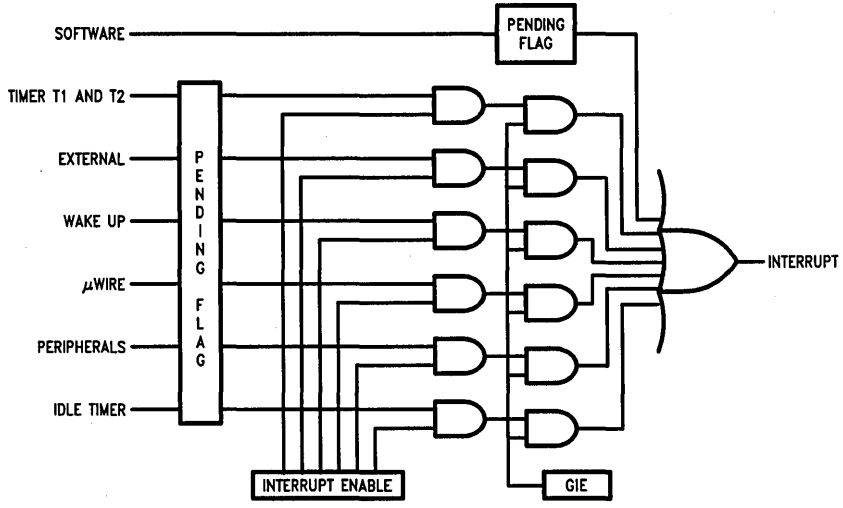
The user must service the WatchDog at least once before the upper limit of the serivce window expires. The WatchDog may not be serviced more than once in every lower limit of the service window. The user may service the WatchDog as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDCNT Register is also counted as a WatchDog service.

The WatchDog has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WatchDog, the logic will pull the WDOUT (G1) pin low for an additional 16 t_c–32 t_c cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CL will stop forcing the WDOUT output low.

The WatchDog service window will restart when the WDOUT pin goes inactive. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WatchDog service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed at $\overline{\text{RESET}}$, but if it powers up low then the WatchDog will time out and disable.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_c–32 t_c clock cycles. The Clock Monitor generates a continual Clock Moni-



TL/DD/9766–18

FIGURE 12. COP888CL Interrupt Block Diagram

WatchDog Operation (Continued)

tor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

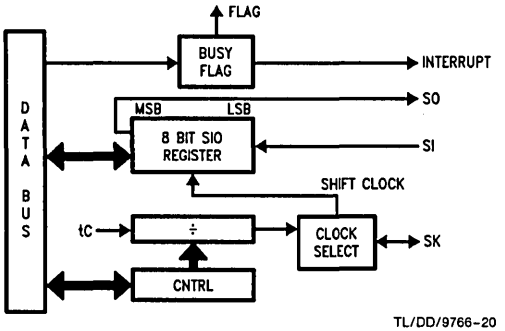
$1/t_C > 10$ kHz—No clock rejection.

$1/t_C < 10$ Hz—Guaranteed clock rejection.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CL to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E^2PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 13* shows a block diagram of the MICROWIRE logic.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.



TL/DD/9766-20

FIGURE 13. MICROWIRE Block Diagram

The CNTRL register is used to configure and control the MICROWIRE mode. To use the MICROWIRE, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table IV details the different clock rates that may be selected.

TABLE III

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WatchDog Output
Mismatch	Don't Care	Don't Care	Error: Generate WatchDog Output
Don't Care	Don't Care	Mismatch	Error: Generate WatchDog Output

TABLE IV

SL1	SL0	SK
0	0	$2 \times t_C$
0	1	$4 \times t_C$
1	x	$8 \times t_C$

Where t_C is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP888CL may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 14* shows how two COP888CL microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock forthe SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CL. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table III summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SKclock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

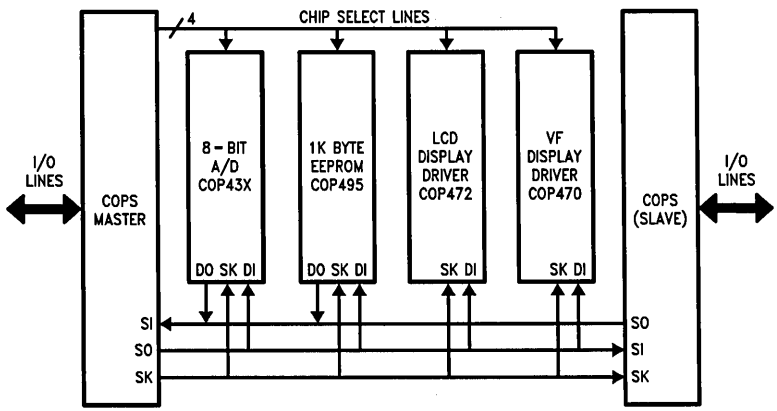
The COP888CL allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock ahd shifted out on the rising edge of the SK clock. In the alternate SK phase mode the SIO register is shifted on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE V

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE Master
0	1	TRI-STATE	Int. SK	MICROWIRE Master
1	0	SO	Ext. SK	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE Slave



TL/DD/9766-21

FIGURE 14. MICROWIRE Application

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space

Address	Contents
00 to 6F	On-Chip RAM bytes
70 to BF	Unused RAM Address Space
C0	Timer T2 Lower Byte
C1	Timer T2 Upper Byte
C2	Timer T2 Autoload Register T2RA Lower Byte
C3	Timer T2 Autoload Register T2RA Upper Byte
C4	Timer T2 Autoload Register T2RB Lower Byte
C5	Timer T2 Autoload Register T2RB Upper Byte
C6	Timer T2 Control Register
C7	WatchDog Service Register (Reg:WDCNT)
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	A/D Converter Control Register (Reg:ENAD)
CC	A/D Converter Result Register (Reg: ADRSLT)
CD to CF	Reserved
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved

Reading memory locations 70-7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

The COP888CL has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CL. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the LBI instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
<–	Loaded with
<–>	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A <− A + Meml
ADC	A,Meml	ADD with Carry	A <− A + Meml + C, C <− Carry HC <− Half Carry
SUBC	A,Meml	Subtract with Carry	A <− A − Meml + C, C <− Carry HC <− Half Carry
AND	A,Meml	Logical AND	A <− A and Meml
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A <− A or Meml
XOR	A,Meml	Logical EXclusive OR	A <− A xor Meml
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF EQual	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A not = Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B not = Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg <− Reg− 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A <−> Mem
LD	A,Meml	LoaD A with Memory	A <− Meml
LD	B,Imm	LoaD B with Immed.	B <− Imm
LD	Mem,Imm	LoaD Memory Immed	Mem <− Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg <− Imm
X	A, [B ±]	EXchange A with Memory [B]	A <−> [B], (B <− B ±1)
X	A, [X ±]	EXchange A with Memory [X]	A <−> [X], (X <− ±1)
LD	A, [B±]	LoaD A with Memory [B]	A <−[B], (B <−B ±1)
LD	A, [X±]	LoaD A with Memory [X]	A <− [X], (X <− X±1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] <−Imm, (B <− B±1)
CLR	A	CLeaR A	A <− 0
INC	A	INCrement A	A<− A + 1
DEC	A	DECrementA	A <− A − 1
LAID		Load A InDirect from ROM	A <− ROM (PU,A)
DCOR	A	Decimal CORrect A	A <− BCD correction (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C −> A7 −> . . . −> A0 −> C
RLC	A	Rotate A Left thru C	C <− A7 <− . . . <− A0 <− C
SWAP	A	SWAP nibbles of A	A7 . . . A4 <−> A3 . . . A0
SC		Set C	C <− 1, HC <− 1
RC		Reset C	C <− 0, HC <− 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP <− SP + 1, A <− [SP]
PUSH	A	PUSH A onto the stack	[SP] <− A, SP <− SP − 1
VIS		Vector to Interrupt Service Routine	PU <− [VU], PL <− [VL]
JMPL	Addr.	Jump absolute Long	PC <− ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9 . . . 0 <− i (i = 12 bits)
JP	Disp.	Jump relative short	PC <− PC + r (r is −31 to +32, not 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] <− PL, [SP−1] <− PU,SP−2, PC <− ii
JSR	Addr	Jump SubRoutine	[SP] <− PL, [SP−1] <− PU,SP−2, PC9 . . . 0 <− i
JID		Jump InDirect	PL <− ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL <−[SP], PU <− [SP−1]
RETSK		RETurn and SKip	SP + 2, PL <−[SP],PU <−[SP−1],Skip <− 1
RETI		RETurn from Interrupt	SP + 2, PL <−[SP],PU <−[SP−1],GIE<− 1
INTR		Generate an Interrupt	[SP] <− PL, [SP−1] <− PU, SP−2, PC <− 0FF
NOP		No OPeration	PC <− PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time (1 μs at 10 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 10 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B, Imm				1/1			(IF B < 16)
LD B, Imm				2/2			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

* = > Memory location addressed by B or X or directly.

COP888CL Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP −15	JP −31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A,[B]	0
JP −14	JP −30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP −13	JP −29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A, #i	IFEQ A,[B]	2
JP −12	JP −28	LD 0F3, # i	DRSZ 0F3	X A, [X−]	X A,[B−]	IFGT A, #i	IFGT A,[B]	3
JP −11	JP −27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A, #i	ADD A,[B]	4
JP −10	JP −26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A, #i	AND A,[B]	5
JP −9	JP −25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A, #i	XOR A,[B]	6
JP −8	JP −24	LD 0F7, # i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	7
JP −7	JP −23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A, #i	IFC	8
JP −6	JP −22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md, #i	IFNE A, #i	IFNC	9
JP −5	JP −21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+], #i	INCA	A
JP −4	JP −20	LD 0FB, # i	DRSZ 0FB	LD A,[X−]	LD A,[B−]	LD [B−], #i	DECA	B
JP −3	JP −19	LD 0FC, # i	DRSZ 0FC	LD Md, #i	JMPL	X A,Md	POPA	C
JP −2	JP −18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP −1	JP −17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B], #i	RET	E
JP −0	JP −16	LD 0FF, # i	DRSZ 0FF	*	*	LD B, #i	RETI	F

COP888CL Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B,#0F	IFBNE 0	JSR x000–x0FF	JMP x000–x0FF	JP +17	INTR	0
IFBIT 1,[B]	*	LD B,#0E	IFBNE 1	JSR x100–x1FF	JMP x100–x1FF	JP +18	JP + 2	1
IFBIT 2,[B]	*	LD B,#0D	IFBNE 2	JSR x200–x2FF	JMP x200–x2FF	JP +19	JP + 3	2
IFBIT 3,[B]	*	LD B,#0C	IFBNE 3	JSR x300–x3FF	JMP x300–x3FF	JP +20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400–x4FF	JMP x400–x4FF	JP +21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500–x5FF	JMP x500–x5FF	JP +22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600–x6FF	JMP x600–x6FF	JP +23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700–x7FF	JMP x700–x7FF	JP +24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800–x8FF	JMP x800–x8FF	JP +25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900–x9FF	JMP x900–x9FF	JP +26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00–xAFF	JMP xA00–xAFF	JP +27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00–xBFF	JMP xB00–xBFF	JP +28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00–xCFF	JMP xC00–xCFF	JP +29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00–xDFF	JMP xD00–xDFF	JP +30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00–xEFF	JMP xE00–xEFF	JP +31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00–xFFF	JMP xF00–xFFF	JP +32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CL mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

```
OPTION 1: CLOCK CONFIGURATION
  = 1     Crystal Oscillator (CKI/10)
              G7 (CKO) is clock generator
              output to crystal/resonator
              CKI is the clock input
  = 2     Single-pin RC controlled
              oscillator (CKI/10)
              G7 is available as a HALT
              restart and/or general purpose
              input

OPTION 2: HALT
  = 1     Enable HALT mode
  = 2     Disable HALT mode

OPTION 3: COP888CL BONDING
  = 1     44-Pin PCC
  = 2     40-Pin DIP
  = 3     28-Pin PCC
  = 4     28-Pin DIP
```

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	TBD	Programmer's Manual		TBD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

Information System

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud modem, and a telephone.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit
Parity: None
Stop Bit

Operation: 24 Hours, 7 Days

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

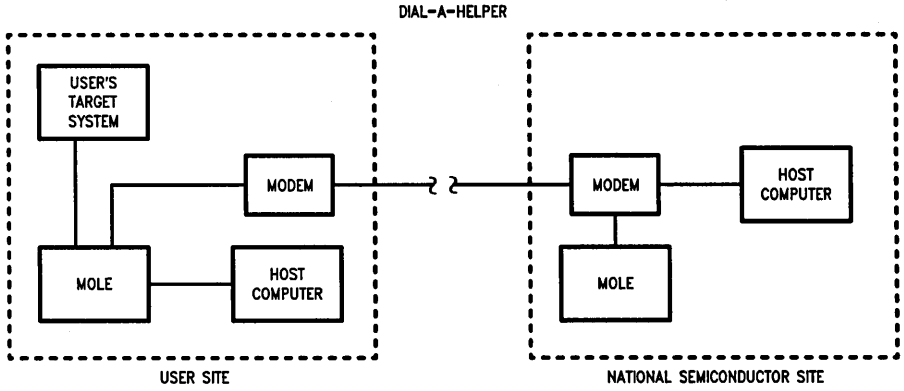
Information System Package Contents
- Dial-A-Helper User Manual P/N
- Public Domain Communications Software

Factory Applications Support

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occuring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will responed to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.



TL/DD/9766-24

National Semiconductor

ADVANCE INFORMATION

COP888CF Single-Chip microCMOS Microcontroller

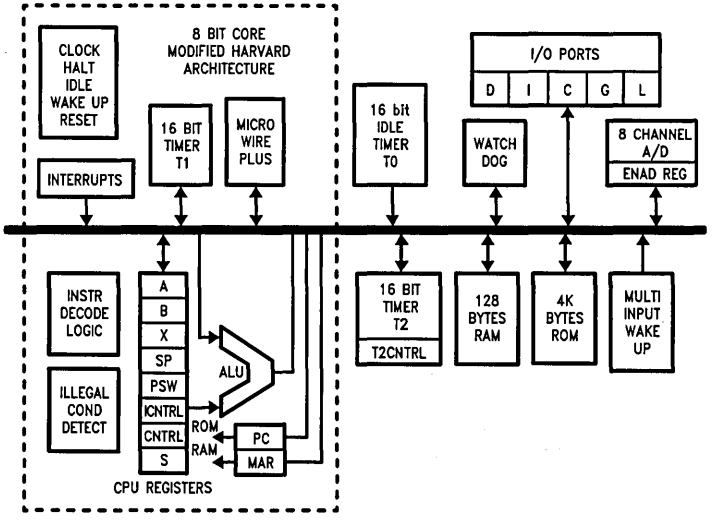
General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. The COP888CF is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 4096 bytes on-board ROM
- 128 bytes on-board RAM
- Single supply operation: 2.5V–6V
- 8-channel A/D converter with prescaler and both differential and single ended modes
- MICROWIRE/PLUS™ serial I/O
- Watch Dog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Timers TA, TB (Each with 2 Interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
- Two 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC or 40 N or 28 N or 28 PCC
  - — 44 PCC with 37 I/O pins
  - — 40 N with 33 I/O pins
  - — 28 PCC or 28 N, each with 21 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Extended temperature range: −55°C to +125°C
- ROMless mode for accurate emulation and external program capability
- Single chip COP8XX piggy back emulation device
- Real time emulation and full program debug offered by National's MOLE™ Development System

Block Diagram



TL/DD/9425–1

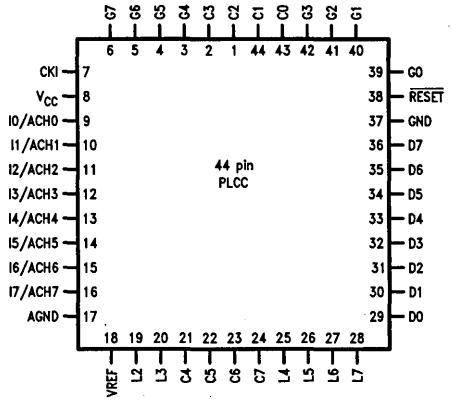
FIGURE 1. COP888CF Block Diagram

General Description (Continued)

It is a fully static part, fabricated using double-metal-silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), an 8-channel, 8-bit A/D converter with both differential and single ended modes, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CF operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The COP888CF may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Connection Diagrams

Plastic Chip Carrier

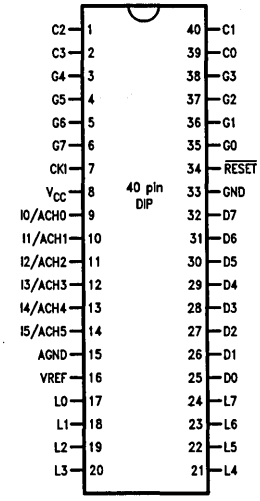


TL/DD/9425-2

Top View

Order Number COP888CF-XXX/V
See NS Plastic Chip Package Number V44A

Dual-In-Line Package

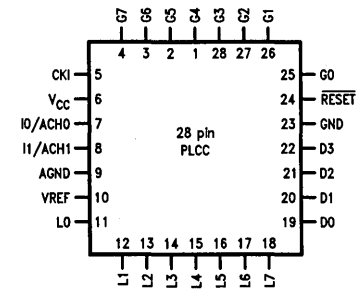


TL/DD/9425-4

Top View

Order Number COP888F-XXX/N
See NS Molded Package Number N40A

Plastic Chip Carrier

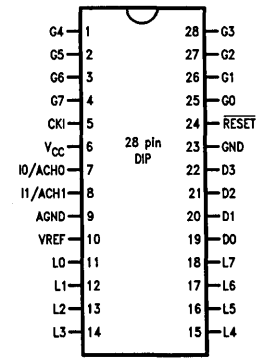


TL/DD/9425-3

Top View

Order Number COP884CF-XXX/V
See NS Plastic Chip Package Number V28A

Dual-In-Line Package



TL/DD/9425-5

Top View

Order Number COP884CF-XXX/N
See NS Molded Package Number N28A

FIGURE 2. COP888CF Connection Diagrams

Connection Diagrams (Continued)

COP888CF Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	—
L1	I/O	MIWU		12	18	—
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	CKO			4	6	6
D0	O	ROM DATA+		19	25	29
D1	O	PCL+		20	26	30
D2	O	EMUL+		21	27	31
D3	O	PCU+		22	28	32
I0	I	ACH0		7	9	9
I1	I	ACH1		8	10	10
I2	I	ACH2		—	11	11
I3	I	ACH3		—	12	12
I4	I	ACH4		—	13	13
I5	I	ACH5		—	14	14
I6	I	ACH6		—	—	15
I7	I	ACH7		—	—	16
D4	O	S CLOCK+			29	33
D5	O	HALTSEL+			30	34
D6	O	LOAD+			31	35
D7	O	D DATA+			32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V_{REF}	$+V_{REF}$			10	16	18
AGND	AGND			9	15	17
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

– = Unbonded Pins

+ = Only in the ROMless Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage				6	V
Power Supply Ripple (Note 1)	Peak-to-Peak	2.5		0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	V_{CC} = 6V, t_c = 1 μs			15	mA
CKI = 4 MHz	V_{CC} = 2.5V, t_c = 2.5 μs			2	mA
HALT Current (Note 3)	V_{CC} = 6V, CKI = 0 MHz		<26		μA
IDLE Current					
CKI = 10 MHz	V_{CC} = 6V, t_c = 1 μs			5	mA
CKI = 4 MHz	V_{CC} = 2.5V, t_c = 2.5 μs			0.6	mA
Input Levels					
Reset					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6V, V_{IN} = 0V	−2		+2	μA
Input Pullup Current	V_{CC} = 6V, V_{IN} = 0V	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4V, V_{OH} = 3.3V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4V, V_{OL} = 1V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.2			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4V, V_{OH} = 2.7V	10		100	μA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	μA
Source (Push-Pull Mode)	V_{CC} = 4V, V_{OH} = 3.3V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink (Push-Pull Mode)	V_{CC} = 4V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			mA
TRI-STATE Leakage		−2		+2	μA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. If the A/D is not being used and minimum standby current is desired, V_{REF} should be tied to AGND (effectively shorting the Reference resistor). The clock monitor is disabled.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $-40°C \leq T_A \leq +85°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup			200		mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)		2		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications $V_{CC} = 5V \pm 10\%$ $(V_{SS} - 0.050V) \leq$ Any Input $\leq (V_{CC} + 0.050V)$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Total Unadjusted Error (Note 5)	V_{REF} = 5V			$\pm$½	LSB
Input Reference Resistance		1.6		4.8	kΩ
Common Mode Input Range				TBD	V
DC Common Mode Error				$\pm$¼	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
Power Supply Sensitivity				$\pm$¼	LSB
A/D Clock Frequency (Note 7)		0.1		1.67	MHz
Conversion Time (Note 6)			12		A/D Clock Cycles

Note 5: Total Unadjusted Error includes offset, full-scale, and multiplexer errors.

Note 6: Conversion Time includes sample and hold time.

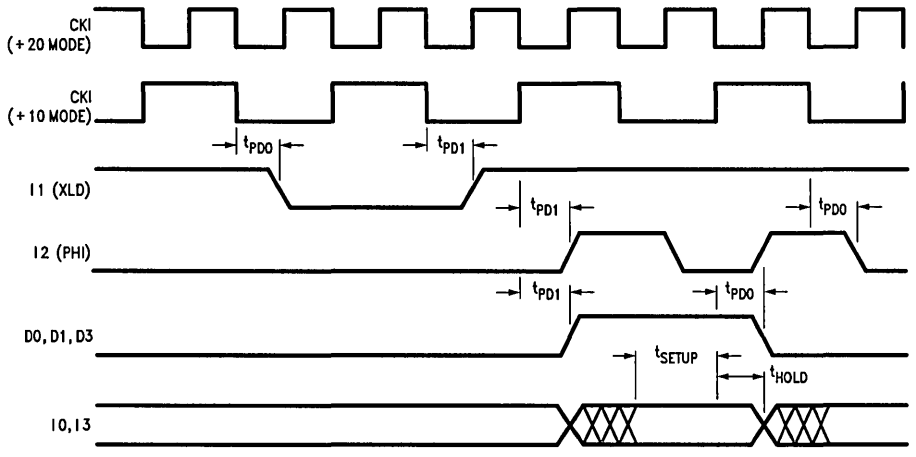
Note 7: See Prescaler description.

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator, or	$4V \le V_{CC} \le 6V$	1		DC	μs
External Oscillator	$2.5V \le V_{CC} < 4V$	2.5		DC	μs
R/C Oscillator	$4V \le V_{CC} \le 6V$	3		DC	μs
	$2.5V \le V_{CC} < 4V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 8)	f_r = Max	40		60	%
Rise Time (Note 8)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 8)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4V \le V_{CC} \le 6V$	200			ns
	$2.5V \le V_{CC} < 4V$	500			ns
t_{HOLD}	$4V \le V_{CC} \le 6V$	60			ns
	$2.5V \le V_{CC} < 4V$	150			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4V \le V_{CC} \le 6V$			0.7	μs
	$2.5V \le V_{CC} < 4V$				
All Others	$4V \le V_{CC} \le 6V$			1	μs
	$2.5V \le V_{CC} < 4V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UV})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

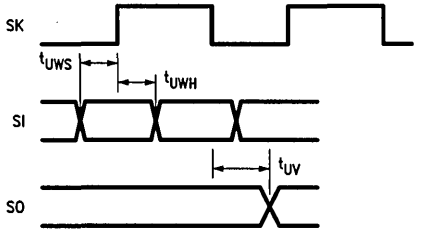
Note 8: Parameter sample but not 100% tested.

AC Electrical Characteristics (Continued)



TL/DD/9425-25

FIGURE 2a. AC Timing Diagrams in ROMless Mode



TL/DD/9425-26

FIGURE 2b. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

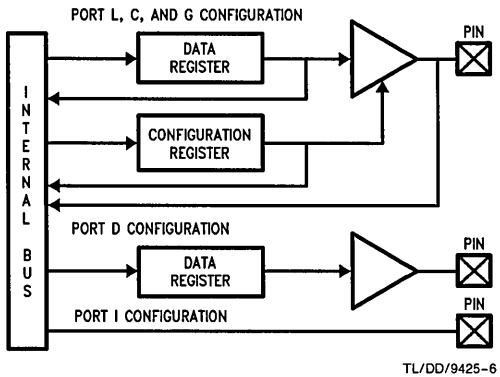
V_{REF} and AGND are the reference voltage pins for the on-board A/D converter.

CKI is the clock input. This can come from an external source, a R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CF contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CF memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CF. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/9425–6

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wakeup (MIWU) on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B. L0 and L1 are not available on the 44-pin version of the COP888CF, since they are replaced by V_{REF} and AGND. L0 and L1 are not terminated on the 44-pin version. Consequently, reading L0 or L1 as inputs will return unreliable data with the 44-pin package, so this data should be masked out with user software when the L port is read for input data.

Port L has the following alternate features:

- L0 MIWU
- L1 MIWU
- L2 MIWU
- L3 MIWU
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU
- L7 MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 serves as the dedicated CKO clock output. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin or general purpose input (R/C clock configuration), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLK Delay	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE™ Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT WatchDog and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

Port I is an 8-bit Hi-Z input port, and also provides the analog inputs to the A/D converter. The 28- and 40-pin devices do not have a full complement of Port I pins. The unavailable pins are not terminated (i.e. they are floating). A read operation from these unterminated pins will return unpredictable values. The user should ensure that the software takes this into account by either masking out these inputs,

Pin Descriptions (Continued)

or else restricting the accesses to bit operations only. If unterminated, Port I pins will draw power only when addressed (i.e. it will be in short spikes).

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CF is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CF architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_C) cycle time.

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)
PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with RESET.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CF consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CF vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE counter). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The COP888CF has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

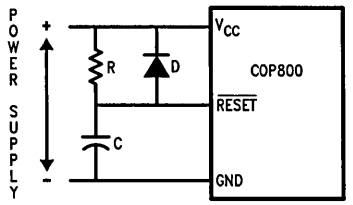
The instruction set of the COP888CF permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CF (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for Ports L, G, and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is initialized high with RESET. The PC, PSW, CNTRL, ICNTRL, and T2CNTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The A/D control register ENAD is cleared, resulting in the ADC being powered down initially. The Stack Pointer, SP, is initialized to 06F Hex.

The COP888CF comes out of RESET with both the WatchDog logic and the Clock Monitor detector armed, and with both the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor detector circuits are inhibited during RESET. The WatchDog service window bits are initialized to the maximum WatchDog service window of 64k t_C clock cycles. The Clock Monitor bit is initialized high, and will cause a Clock Monitor error following RESET if the clock has not reached the minimum specified frequency at the termination of RESET. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16–32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 4* should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes. It is recommended that the components of the RC network be selected to provide a RESET delay of at least five times the power supply rise time or the minimum RESET pulse width, whichever is greater.



TL/DD/9425–7

RC > 5 × Power Supply Rise Time

FIGURE 4. Recommended RESET Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

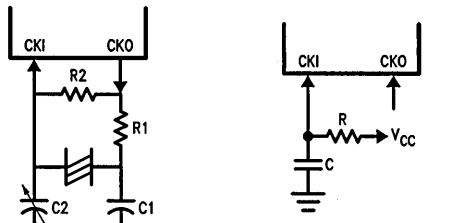


FIGURE 5. Crystal and R/C Oscillator Diagrams

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. DC reference current contribution from the A/D converter—I6

Thus the total current drain, It, is given as

$$It = I1 + I2 + I3 + I4 + I5 + I6$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Some sample current drain values at V_{CC} = 5V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)	—	0.005

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE control register contains the following bits:

SL1 & SL0	Select the MICROWIRE clock divide by (00 = 2, 01 = 4, 1× = 8)
IEDG	External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
MSEL	Selects G5 and G4 as MICROWIRE signals SK and SO respectively
T1C0	Timer T1 Start/Stop control in timer modes 1 and 2
	Timer T1 Underflow Interrupt Pending Flag in timer mode 3
T1C1	Timer T1 mode control bit
T1C2	Timer T1 mode control bit
T1C3	Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0

Bit 7 Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt
BUSY	MICROWIRE busy shifting flag
EXPND	External interrupt pending
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
T1PNDA	Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
C	Carry Flag
HC	Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE

Bit 7 Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

Control Registers (Continued)

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

- T1ENB Timer T1 Interrupt Enable for T1B Input capture edge
- T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge
- μWEN Enable MICROWIRE/PLUS interrupt
- μWPND MICROWIRE/PLUS interrupt pending
- T0EN Timer T0 Interrupt Enable (Bit 12 toggle)
- T0PND Timer T0 Interrupt pending
- LPEN L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

  Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PNDB	T1ENB
Bit 7							Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

- T2ENB Timer T2 Interrupt Enable for T2B Input capture edge
- T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge
- T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
- T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
- T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3
- T2C1 Timer T2 mode control bit
- T2C2 Timer T2 mode control bit
- T2C3 Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

Emulation and ROMless Modes

The COP888CF can address up to 32 kbytes of address space. If at power up, D2 is held at ground, the COP888CF executes from external memory. Port D is used to interface to external program memory. The address comes out in a serial fashion and the data from the external program memory is read back in a serial fashion. The Port D pins perform the following functions.

- D0 Shifts in ROM data
- D1 Shifts out lower eight bits of PC
- D2 Places the μC in the ROMless mode if grounded at reset
- D3 Shifts out upper eight bits of PC
- D4 Data Shift Clock
- D5 HALT Mask Option select pin (D5 = 0) for HALT enable, D5 = 1 for HALT disable)
- D6 Load Clock
- D7 Shifts out recreated Port D data

The most significant bit of the data to come out on the D3 pin is a status signal.. It is used by the MOLE development system. This "lost" output port (D0–D7) can be accurately reconstructed with external components as shown in *Figure 6*, providing an accurate emulation.

The 44-pin and 40-pin versions of the COP888CF have a full complement of the D Port pins and can be used in the ROMless mode. However, it should be noted that the 44-pin device can only emulate itself and not the 40-pin or 28-pin devices as it has only 6 Port L pins while the other two devices have a full complement of Port L pins.

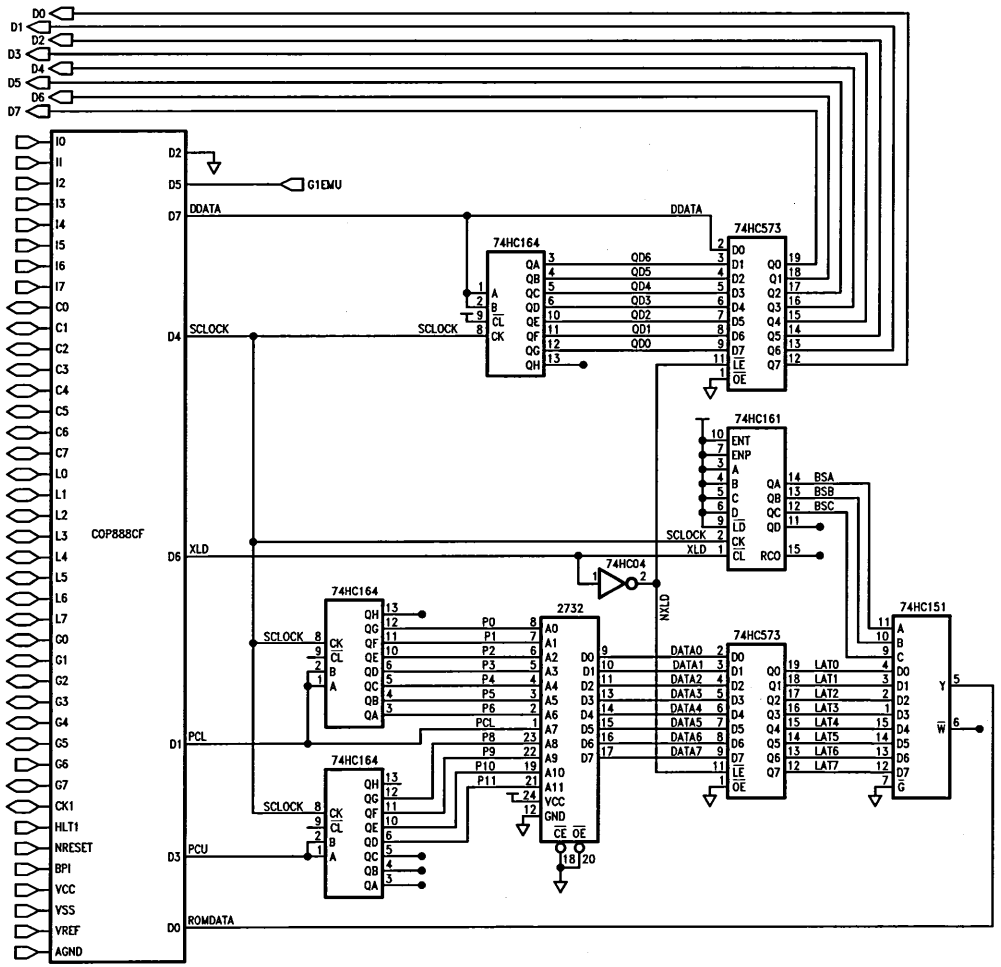
The 28-pin part cannot be used for emulation since it does not have the full complement of 8 D Port pins necessary for entering the ROMless mode.

Note that in the ROMless mode the D Port is recreated one full clock cycle behind the normal port timings.

Note: Standard parts used in the ROMless mode will operate only at a reduced frequency (to be defined).

The COP888CF device has a spare D pin (D5) in the emulation mode since only seven pins are required for emulation and recreation. This pin D5 is used in the emulation mode to enable or disable the HALT mask option feature.

Figure 6 shows the COP888CF Emulation Mode Schematic.



TL/DD/9425–10

FIGURE 6. COP888CF Emulation Mode Schematic

Power Save Modes

The COP888CF offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CF is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock, timers, and A/D converter, are stopped. The WatchDog logic on the COP888CF is disabled during the HALT mode. However, the clock monitor circuitry remains active. In the HALT mode, the power requirements of the COP888CF are minimal and the applied voltage (V_{CC}) may be decreased to V_r (V_r = 2.0V) without altering the state of the machine.

The COP888CF supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method preludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET input low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE time is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared at reset.

The COP88CF has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CF will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CF cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit remains active during HALT mode in order to ensure a clock monitor error if the COP888CF inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CF is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the WatchDog logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

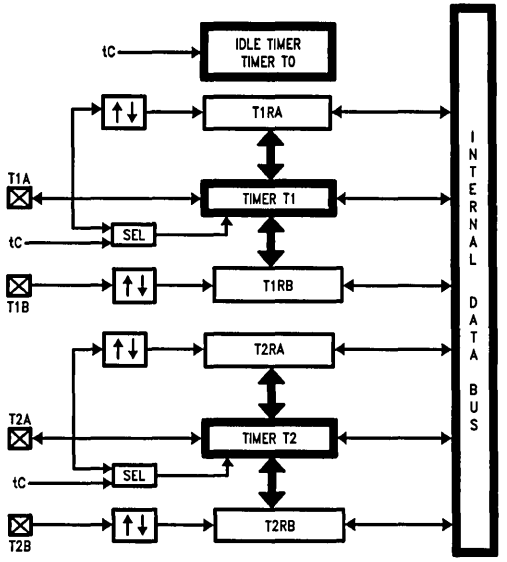
As with the HALT mode, the COP888CF can be returned to normal operation with a RESET, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the twelfth bit (representing 4.096 ms at internal clock frequency of 1 MHz (t_c = 1 μs)) of the IDLE Timer toggles.

This toggle condition of the twelfth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

The user has the option of being interrupted with a transition on the twelfth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the COP888CF will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CF will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.



TL/DD/9425–11

FIGURE 7. Timers for the COP888CF

Timers

The COP888CF contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

Figure 7 shows a block diagram for the timers on the COP888CF.

TIMER T0 (IDLE TIMER)

The COP888CF supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_C. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)
WatchDog logic (See WatchDog description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the twelfth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency (t_C = 1 μs). A control flag T0EN allows the interrupt from the twelfth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1 AND TIMER T2

The COP888CF has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2, are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CF to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CF to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode.

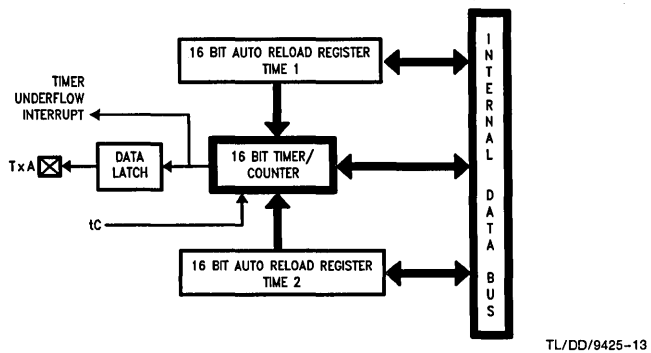
The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.



TL/DD/9425–13

FIGURE 8. Timer in PWM Mode

Timers (Continued)

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

Mode 3. Input Capture Mode

The COP888CF can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_C rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

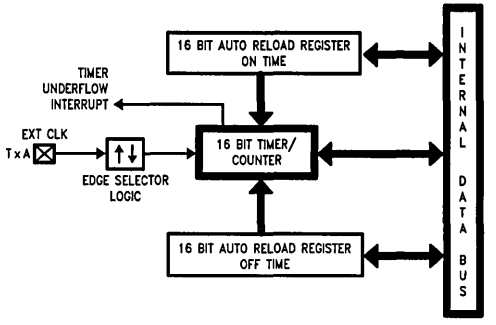
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

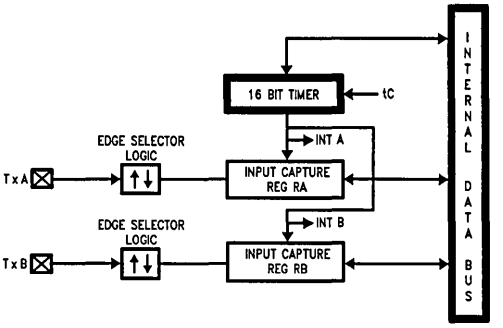
The timers T1 and T2 have indentical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control



TL/DD/9425–14

FIGURE 9. Timer in External Event Counter Mode



TL/DD/9425–15

FIGURE 10. Timer in Input Capture Mode

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_C
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_C
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C

Detection of Illegal Conditions

The COP888CF will detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during RESET. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F Hex is read as all 1's, which in turn will cause the program to return to address FFFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

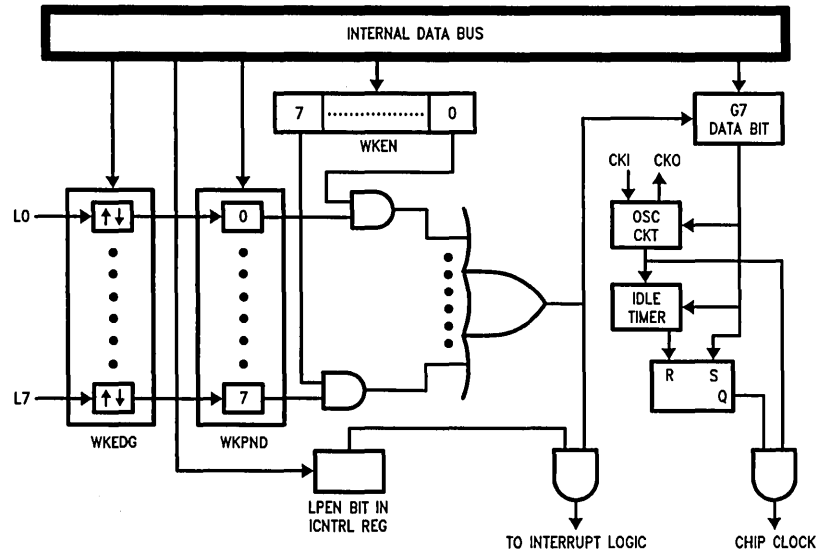
a. Executing from undefined ROM
b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following RESET, but might not contain the same program initialization procedures).

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CF from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Multi-Input Wakeup (Continued)



TL/DD/9425–16

FIGURE 11. Multi-Input Wake Up Logic

Figure 11 shows the Multi-Input Wakeup logic for the COP888CF microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CF to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following RESET, since the L port inputs are left floating as a result of RESET.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called Reg: WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since the Reg: WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CF will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

All three registers Reg:WKEN, Reg:WKPND and Reg:WKEDG are read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG

Multi-Input Wakeup (Continued)

specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CF out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CF will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CF will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CF to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during RESET, so the clock start up delay is not present following RESET with the RC clock options.

A/D Converter

The COP888CF contains an 8-channel, multiplexed input, successive approximation, A/D converter. Two dedicated pins, V_{REF} and AGND are provided for voltage reference.

OPERATING MODES

The A/D converter supports ratiometric measurements. It supports both Single Ended and Differential modes of operation.

Four specific analog channel selection modes are supported. These are as follows:

Allow any specific channel to be selected at one time. The A/D converter performs the specific conversion requested and stops.

Allow any specific channel to be scanned continuously. In other words, the user will specify the channel and the A/D converter will keep on scanning it continuously. The user can come in at any arbitrary time and immediately read the result of the last conversion. The user does not have to wait for the current conversion to be completed.

Allow any differential channel pair to be selected at one time. The A/D converter performs the specific differential conversion requested and stops.

Allow any differential channel pair to be scanned continuously. In other words, the user will specify the differential channel pair and the A/D converter will keep on scanning it continuously. The user can come in at any arbitrary time and immediately read the result of the last differential conversion. The user does not have to wait for the current conversion to be completed.

The A/D converter is supported by two memory mapped registers, the result register and the mode control register. When the COP888CF is reset, the control register is cleared and the A/D is powered down in the single ended conversion mode. The A/D result register has unknown data following reset.

A/D Control Register

A control register, Reg: ENAD, contains 3 bits for channel selection, 3 bits for prescaler selection, and 2 bits for mode selection. An A/D conversion is initiated by writing to the ENAD control register. The result of the conversion is available to the user from the A/D result register, Reg: ADRSLT.

Reg: ENAD

CHANNEL SELECT	MODE SELECT	PRESCALER SELECT
Bits 7, 6, 5	Bits 4,3	Bits 2, 1, 0

CHANNEL SELECT

This 3-bit field is used to specify the channel address to select one of the 8 A/D channels in Single Ended mode, or select one of the 4 A/D channel pairs in the Differential mode.

Single Ended mode:

Bit 7	Bit 6	Bit 5	Channel No.
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

A/D Converter (Continued)

Differential mode:

Bit 7	Bit 6	Bit 5	Channel Pairs (+. −)
0	0	0	0, 1
0	0	1	1, 0
0	1	0	2, 3
0	1	1	3, 2
1	0	0	4, 5
1	0	1	5, 4
1	1	0	6, 7
1	1	1	7, 6

MODE SELECT

This 2-bit field is used to select the mode of operation (single conversion, continuous conversions, differential, single ended) as shown in the following table.

Bit 4	Bit 3	Mode
0	0	Single Ended mode, single conversion
0	1	Single Ended mode, continuous scan of a single channel into the result register
1	0	Differential mode, single conversion
1	1	Differential mode, continuous scan of a channel pair into the result register

PRESCALER SELECT

This 3-bit field is used to select one of the seven prescaler clocks for the A/D converter. The prescaler also allows the A/D clock inhibit power saving mode to be selected. The following table shows the various prescaler options.

Bit 2	Bit 1	Bit 0	Clock Select
0	0	0	Inhibit A/D clock
0	0	1	Divide by 1
0	1	0	Divide by 2
0	1	1	Divide by 4
1	0	0	Divide by 6
1	0	1	Divide by 12
1	1	0	Divide by 8
1	1	1	Divide by 16

ADC Operation

The A/D converter interface works as follows. Writing to the A/D control register ENAD initiates an A/D conversion unless the prescaler value is set to 0, in which case the ADC clock is stopped and the ADC is powered down. The conversion sequence starts at the beginning of the write to ENAD operation by deselecting and powering up the ADC. At the first falling edge of the converter clock following the write operation (not counting the falling edge if it occurs at the same time as the write operation ends), the sample signal turns on for two clock cycles. The ADC is selected in the middle of the sample period. If the ADC is in single conversion mode, the conversion complete signal from the ADC will generate a power down for the A/D converter. If the ADC is in continuous mode, the conversion complete signal will restart the conversion sequence by deselecting the ADC for one converter clock cycle before starting the next sample. The ADC 8-bit result is loaded into the A/D result register (ADRSLT) except during LOAD clock high, which prevents transient data (resulting from the ADC writing a new result over an old one) being read from ADRSLT.

PRESCALER

The COP888CF A/D Converter (ADC) contains a prescaler option which allows seven different clock selections. The A/D clock frequency is equal to CKI divided by the prescaler value. Note that the prescaler value must be chosen such that the A/D clock falls within the specified range. With a prescaler of 6 selected, the maximum A/D clock frequency is 1.67 MHz (10 MHz divided by 6). This equates to a 600 ns ADC clock cycle.

The A/D converter takes 12 ADC clock cycles to complete a conversion. Thus the minimum ADC conversion time for the COP888CF is 7.2 μs when a prescaler of 6 has been selected. These 12 ADC clock cycles necessary for a conversion consist of 1 cycle at the beginning for reset, 2 cycles for sampling, 8 cycles for converting, and 1 cycle for loading the result into the COP888CF A/D result register (ADRSLT). This A/D result register is a read-only register. The COP888CF cannot write into ADRSLT.

The prescaler also allows an A/D clock inhibit option, which saves power by powering down the A/D when it is not in use.

Note: The A/D converter is also powered down when the COP888CF is in either the HALT or IDLE modes. If the ADC is running when the COP888CF enters the HALT or IDLE modes, the ADC will power down during the HALT or IDLE, and then will reinitialize the conversion when the COP888CF comes out of the HALT or IDLE modes.

Interrupts

The COP888CF supports a vectored interrupt scheme. It supports a total of ten interrupt sources. The following table lists all the possible COP888CF interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_C cycles to execute.

Interrupts (Continued)

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	T0 Bit 12 Toggle	0yF8–0yF9
(4)	Timer T1	T1 Underflow/ T1A Capture Edge	0yF6–0yF7
(5)	Timer T1	T1B Capture Edge	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
	Reserved	for UART	0yEE–0yEF
	Reserved	for UART	0yEC–0yED
(7)	Timer T2	T2 Underflow/ T2A Capture Edge	0yEA–0yEB
(8)	Timer T2	T2B Capture Edge	0yE8–0yE9
	Reserved	for Future Use	0yE6–0yE7
	Reserved	for Future Use	0yE4–0yE5
(9)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(10) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 12 shows the COP888CF Interrupt block diagram.

Interrupts (Continued)

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to RESET, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit **(not accessible by the user)** is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction.

It is cleared by RESET and by the RPND instruction.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

The COP888CF contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

WatchDog

The COP888CF WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDCNT which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table I shows the WDCNT register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 ad 6 of the WDCNT register allow the user to pick an upper limit of the service window.

Table II shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDCNT register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDCNT Register is the Clock Monitor Select bit.

TABLE I

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE II

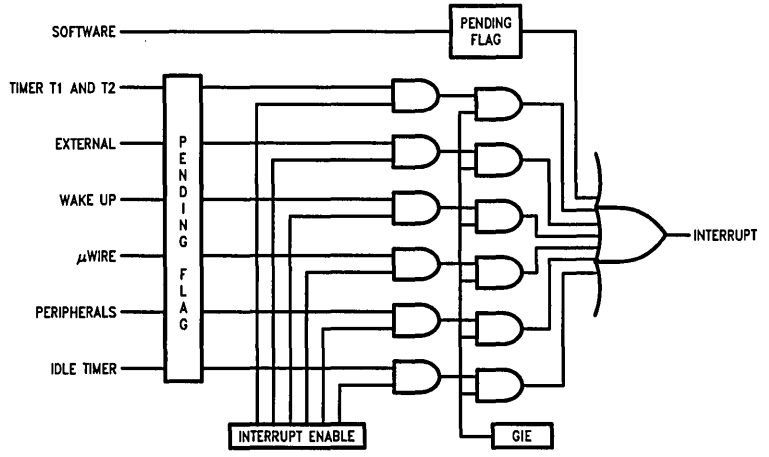
WDCNT Bit 7	WDCNT Bit 6	Service Window (Lower-Upper Limits)
0	0	2k-8k t_C Cycles
0	1	2k-16k t_C Cycles
1	0	2k-32k t_C Cycles
1	1	2k-64k t_C Cycles

Clock Monitor

The Clock Monitor aboard the COP888CF can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_C$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WatchDog Operation

The WatchDog and Clock Monitor are disabled during $\overline{\text{RESET}}$. The COP888CF comes out of $\overline{\text{RESET}}$ with the WatchDog armed, the WatchDog Window Select bits (bits 6,



TL/DD/9425–18

FIGURE 12. COP888CF Interrupt Block Diagram

WatchDog Operation (Continued)

7 of the WDCNT Register) set, and the Clock Monitor bit (bit 0 of the WDCNT Register) enabled. Thus, a Clock Monitor error will occur after coming out of RESET, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDCNT register can be written to only once after RESET and the key data (bits 5 through 1 of the WDCNT Register) must match to be a valid write. This write to the WDCNT register involves two irrevocable choices: (i) the selection of the WatchDog service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDCNT Register involves selecting or deselecting the Clock Monitor, select the WatchDog service window and match the WatchDog key data. Subsequent writes to the WDCNT register will compare the value being written by the user to the WatchDog service window value and the key data (bits 7 through 1) in the WDCNT Register. Table III shows the sequence of events that can occur.

The user must service the WatchDog at least once before the upper limit of the serivce window expires. The WatchDog may not be serviced more than once in every lower limit of the service window. The user may service the WatchDog as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDCNT Register is also counted as a WatchDog service.

The WatchDog has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WatchDog, the logic will pull the WDOUT (G1) pin low for an additional 16 t_C–32 t_C cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CF will stop forcing the WDOUT output low.

The WatchDog service window will restart when the WDOUT pin goes inactive. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WatchDog service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed at RESET, but if it powers up low then the WatchDog will time out and disable.

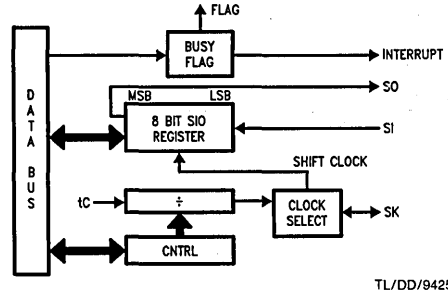
The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_C–32 t_C clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_C > 10$ kHz—No clock rejection.

$1/t_C < 10$ Hz—Guaranteed clock rejection.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CF to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E^2PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 13* shows a block diagram of the MICROWIRE logic.



TL/DD/9425–20

FIGURE 13. MICROWIRE Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE mode. To use the MICROWIRE, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table IV details the different clock rates that may be selected.

TABLE III

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WatchDog Output
Mismatch	Don't Care	Don't Care	Error: Generate WatchDog Output
Don't Care	Don't Care	Mismatch	Error: Generate WatchDog Output

TABLE IV

SL1	SL0	SK
0	0	$2 \times t_C$
0	1	$4 \times t_C$
1	x	$8 \times t_C$

Where t_C is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP888CF may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 14* shows how two COP888CF microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock forthe SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CF. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table III summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SKclock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

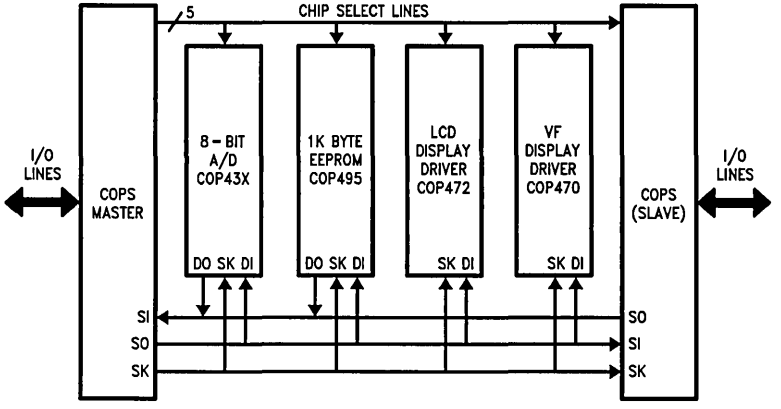
The COP888CF allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock ahd shifted out on the rising edge of the SK clock. In the alternate SK phase mode the SIO register is shifted on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE V

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE Master
0	1	TRI-STATE	Int. SK	MICROWIRE Master
1	0	SO	Ext. SK	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE Slave



TL/DD/9425-21

FIGURE 14. MICROWIRE Application

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space

Address	Contents
00 to 6F	On-Chip RAM bytes
70 to BF	Unused RAM Address Space
C0	Timer T2 Lower Byte
C1	Timer T2 Upper Byte
C2	Timer T2 Autoload Register T2RA Lower Byte
C3	Timer T2 Autoload Register T2RA Upper Byte
C4	Timer T2 Autoload Register T2RB Lower Byte
C5	Timer T2 Autoload Register T2RB Upper Byte
C6	Timer T2 Control Register
C7	WatchDog Service Register (Reg:WDCNT)
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	A/D Converter Control Register (Reg:ENAD)
CC	A/D Converter Result Register (Reg: ADRSLT)
CD to CF	Reserved
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved

Reading memory locations 70-7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

The COP888CF has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CF. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the LBI instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
<–	Loaded with
<–>	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A <− A + Meml
ADC	A,Meml	ADD with Carry	A <− A + Meml + C, C <− Carry HC <− Half Carry
SUBC	A,Meml	Subtract with Carry	A <− A − Meml + C, C <− Carry HC <− Half Carry
AND	A,Meml	Logical AND	A <− A and Meml
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A <− A or Meml
XOR	A,Meml	Logical EXclusive OR	A <− A xor Meml
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF EQual	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A not = Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B not = Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg <− Reg− 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A <−> Mem
LD	A,Meml	LoaD A with Memory	A <− Meml
LD	B,Imm	LoaD B with Immed.	B <− Imm
LD	Mem,Imm	LoaD Memory Immed	Mem <− Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg <− Imm
X	A, [B ±]	EXchange A with Memory [B]	A <−> [B], (B <− B ±1)
X	A, [X ±]	EXchange A with Memory [X]	A <−> [X], (X <− ±1)
LD	A, [B±]	LoaD A with Memory [B]	A <−[B], (B <−B ±1)
LD	A, [X±]	LoaD A with Memory [X]	A <− [X], (X <− X±1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] <−Imm, (B <− B±1)
CLR	A	CLeaR A	A <− 0
INC	A	INCrement A	A<− A + 1
DEC	A	DECrementA	A <− A − 1
LAID		Load A InDirect from ROM	A <− ROM (PU,A)
DCOR	A	Decimal CORrect A	A <− BCD correction (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C −> A7 −> . . . −> A0 −> C
RLC	A	Rotate A Left thru C	C <− A7 <− . . . <− A0 <− C
SWAP	A	SWAP nibbles of A	A7 . . . A4 <−> A3 . . . A0
SC		Set C	C <− 1, HC <− 1
RC		Reset C	C <− 0, HC <− 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP <− SP + 1, A <− [SP]
PUSH	A	PUSH A onto the stack	[SP] <− A, SP <− SP − 1
VIS		Vector to Interrupt Service Routine	PU <− [VU], PL <− [VL]
JMPL	Addr.	Jump absolute Long	PC <− ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9 . . . 0 <− i (i = 12 bits)
JP	Disp.	Jump relative short	PC <− PC + r (r is −31 to +32, not 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] <− PL, [SP−1] <− PU,SP−2, PC <− ii
JSR	Addr	Jump SubRoutine	[SP] <− PL, [SP−1] <− PU,SP−2, PC9 . . . 0 <− i
JID		Jump InDirect	PL <− ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL <−[SP], PU <− [SP−1]
RETSK		RETurn and SKip	SP + 2, PL <−[SP],PU <−[SP−1],Skip <− 1
RETI		RETurn from Interrupt	SP + 2, PL <−[SP],PU <−[SP−1],GIE<− 1
INTR		Generate an Interrupt	[SP] <− PL, [SP−1] <− PU, SP−2, PC <− 0FF
NOP		No OPeration	PC <− PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time (1 μs at 10 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 10 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B, Imm				1/1			(IF B < 16)
LD B, Imm				2/2			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

* = > Memory location addressed by B or X or directly.

COP888CF Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP −15	JP −31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	0
JP −14	JP −30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP −13	JP −29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	2
JP −12	JP −28	LD 0F3, # i	DRSZ 0F3	X A, [X−]	X A,[B−]	IFGT A,#i	IFGT A,[B]	3
JP −11	JP −27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	4
JP −10	JP −26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	5
JP −9	JP −25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	6
JP −8	JP −24	LD 0F7, # i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	7
JP −7	JP −23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	8
JP −6	JP −22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	9
JP −5	JP −21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+],#i	INCA	A
JP −4	JP −20	LD 0FB, # i	DRSZ 0FB	LD A,[X−]	LD A,[B−]	LD [B−],#i	DECA	B
JP −3	JP −19	LD 0FC, # i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	C
JP −2	JP −18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP −1	JP −17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B],#i	RET	E
JP −0	JP −16	LD 0FF, # i	DRSZ 0FF	*	*	LD B,#i	RETI	F

COP888CF Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B,#0F	IFBNE 0	JSR x000–x0FF	JMP x000–x0FF	JP +17	INTR	0
IFBIT 1,[B]	*	LD B,#0E	IFBNE 1	JSR x100–x1FF	JMP x100–x1FF	JP +18	JP + 2	1
IFBIT 2,[B]	*	LD B,#0D	IFBNE 2	JSR x200–x2FF	JMP x200–x2FF	JP +19	JP + 3	2
IFBIT 3,[B]	*	LD B,#0C	IFBNE 3	JSR x300–x3FF	JMP x300–x3FF	JP +20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400–x4FF	JMP x400–x4FF	JP +21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500–x5FF	JMP x500–x5FF	JP +22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600–x6FF	JMP x600–x6FF	JP +23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700–x7FF	JMP x700–x7FF	JP +24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800–x8FF	JMP x800–x8FF	JP +25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900–x9FF	JMP x900–x9FF	JP +26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00–xAFF	JMP xA00–xAFF	JP +27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00–xBFF	JMP xB00–xBFF	JP +28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00–xCFF	JMP xC00–xCFF	JP +29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00–xDFF	JMP xD00–xDFF	JP +30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00–xEFF	JMP xE00–xEFF	JP +31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00–xFFF	JMP xF00–xFFF	JP +32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CF mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

```
OPTION 1: CLOCK CONFIGURATION
  = 1     Crystal Oscillator (CKI/10)
              G7 (CKO) is clock generator
              output to crystal/resonator
              CKI is the clock input
  = 2     Single-pin RC controlled
              oscillator (CKI/10)
              G7 is available as a HALT
              restart and/or general purpose
              input

OPTION 2: HALT
  = 1     Enable HALT mode
  = 2     Disable HALT mode

OPTION 3: COP888CF BONDING
  = 1     44-Pin PLCC
  = 2     40-Pin DIP
  = 3     28-Pin PLCC
  = 4     28-Pin DIP
```

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	TBD	Programmer's Manual		TBD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

Information System

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud modem, and a telephone.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit
Parity: None
Stop Bit

Operation: 24 Hours, 7 Days

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

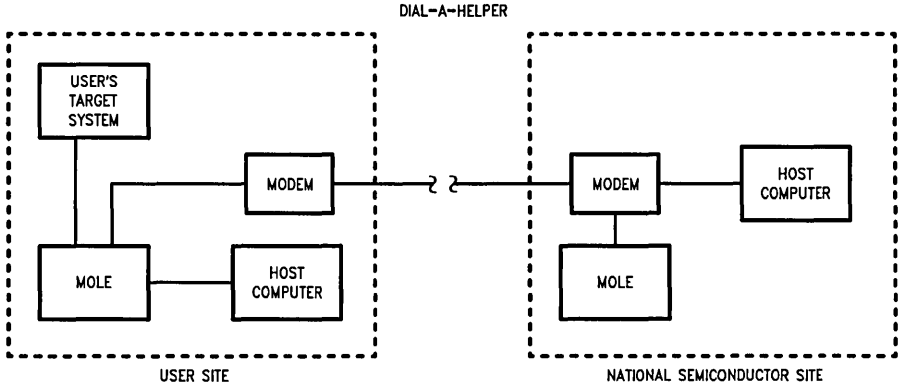
Information System Package Contents
- Dial-A-Helper User Manual
- Public Domain Communications Software

Factory Applications Support

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occuring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will responed to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.



TL/DD/9425-24

National Semiconductor

ADVANCE INFORMATION

COP888CG Single-Chip microCMOS Microcontroller

General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M²CMOS™ process technology. The COP888CG is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 4096 bytes on-board ROM
- 192 bytes on-board RAM
- Single supply operation: 2.5V–6V
- Full duplex UART
- Two comparators
- MICROWIRE/PLUS™ serial I/O
- Watch Dog and Clock Monitor logic
- Idle Timer
- Three 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Fourteen multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Timers (6)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — UART (2)
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC or 40 N or 28 N or 28 PCC
  - — 44 PCC with 39 I/O pins
  - — 40 N with 35 I/O pins
  - — 28 PCC or 28 N, each with 23 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Extended temperature range: −55°C to +125°C
- ROMless mode for accurate emulation and external program capability
- Single chip COP8XX piggy back emulation device
- Real time emulation and full program debug offered by National's MOLE™ Development System

Block Diagram

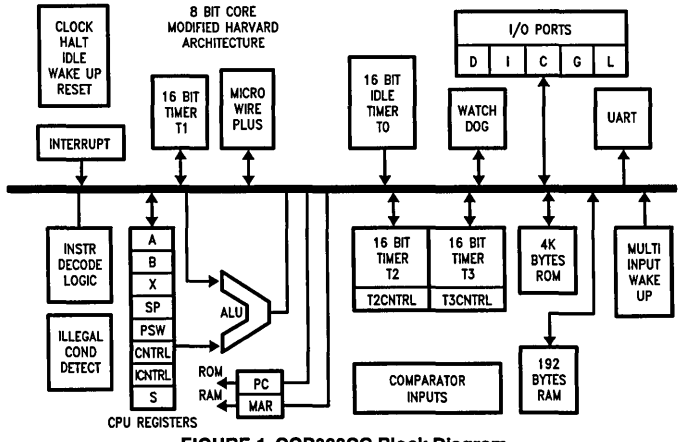


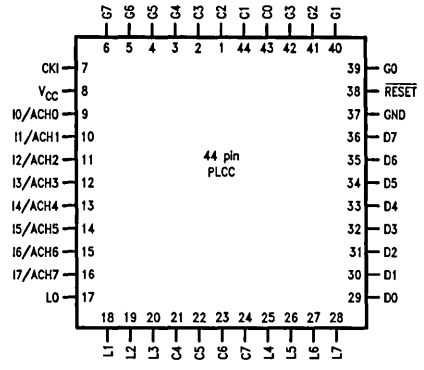
FIGURE 1. COP888CG Block Diagram

General Description (Continued)

It is a fully static part, fabricated using double-metal-silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CG operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The COP888CG may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Connection Diagrams

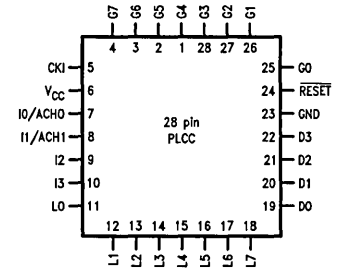
Plastic Chip Carrier



TL/DD/9765-2

Top View

Order Number COP888CG-XXX/V
See NS Plastic Chip Package Number V44A
Plastic Chip Carrier

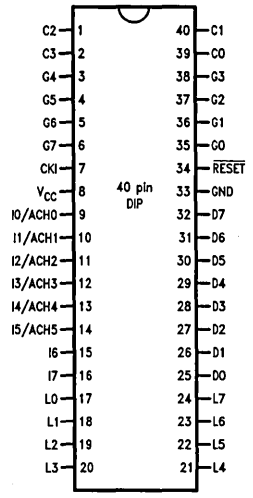


TL/DD/9765-3

Top View

Order Number COP884CG-XXX/V
See NS Plastic Chip Package Number V28A

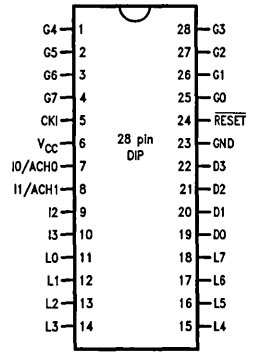
Dual-In-Line Package



TL/DD/9765-4

Top View

Order Number COP888G-XXX/N
See NS Molded Package Number N40A
Dual-In-Line Package



TL/DD/9765-5

Top View

Order Number COP884CG-XXX/N
See NS Molded Package Number N28A

FIGURE 2a. COP888CG Connection Diagrams

Connection Diagrams (Continued)

COP888CG Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU	T3A	17	23	27
L7	I/O	MIWU	T3B	18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	CKO/I			4	6	6
D0	O	ROM DATA+		19	25	29
D1	O	PCL+		20	26	30
D2	O	EMUL+		21	27	31
D3	O	PCU+		22	28	32
I0	I			7	9	9
I1	I	COMP1IN−		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I	COMP2IN−		—	13	13
I5	I	COMP2IN+		—	14	14
I6	I	COMP2OUT		—	15	15
I7	I			—	16	16
D4	O	S CLOCK+			29	33
D5	O	HALTSEL+			30	34
D6	O	LOAD+			31	35
D7	O	D DATA+			32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

− = Unbonded Pins
+ = Only in the ROMless Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics −40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage				6	V
Power Supply Ripple (Note 1)	Peak-to-Peak	2.5		0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	V_{CC} = 6V, t_c = 1 μs			15	mA
CKI = 4 MHz	V_{CC} = 2.5V, t_c = 2.5 μs			2	mA
HALT Current (Note 3)	V_{CC} = 6V, CKI = 0 MHz		<1		μA
IDLE Current					
CKI = 10 MHz	V_{CC} = 6V, t_c = 1 μs			5	mA
CKI = 4 MHz	V_{CC} = 2.5V, t_c = 2.5 μs			0.6	mA
Input Levels					
Reset					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6V, V_{IN} = 0V	−2		+2	μA
Input Pullup Current	V_{CC} = 6V, V_{IN} = 0V	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4V, V_{OH} = 3.3V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4V, V_{OL} = 1V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.2			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4V, V_{OH} = 2.7V	10		100	μA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	μA
Source (Push-Pull Mode)	V_{CC} = 4V, V_{OH} = 3.3V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink (Push-Pull Mode)	V_{CC} = 4V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			mA
TRI-STATE Leakage		−2		+2	μA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The user must disable the clock monitor and the comparators.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup			200		mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)		2		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

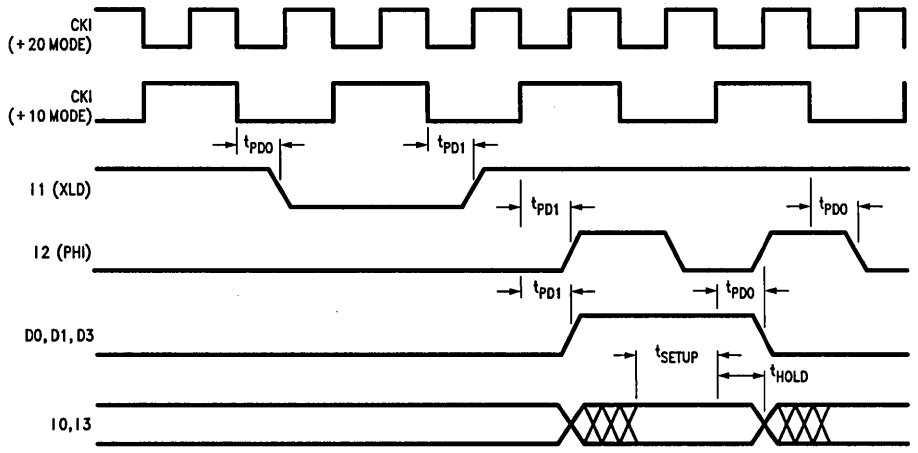
AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator, or	$4V \le V_{CC} \le 6V$	1		DC	μs
External Oscillator	$2.5V \le V_{CC} < 4V$	2.5		DC	μs
R/C Oscillator	$4V \le V_{CC} \le 6V$	3		DC	μs
	$2.5V \le V_{CC} < 4V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 5)	f_r = Max	40		60	%
Rise Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4V \le V_{CC} \le 6V$	200			ns
	$2.5V \le V_{CC} < 4V$	500			ns
t_{HOLD}	$4V \le V_{CC} \le 6V$	60			ns
	$2.5V \le V_{CC} < 4V$	150			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4V \le V_{CC} \le 6V$			0.7	μs
	$2.5V \le V_{CC} < 4V$				
All Others	$4V \le V_{CC} \le 6V$			1	μs
	$2.5V \le V_{CC} < 4V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UV})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

Note 5: Parameter sampled but not 100% tested.

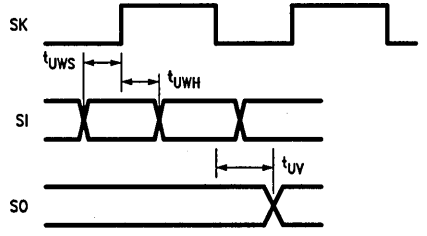
Comparators AC and DC Characteristics V_{CC} = 5V, T_A = 25°C

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \le V_{IN} \le V_{CC} - 1.5V$		10	25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	V_{OL} = 0.4V	1.6			mA
High Level Output Current	V_{OH} = 4.6V	1.6			mA
DC Supply Current (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



TL/DD/9765-6

FIGURE 2b. AC Timing Diagrams in ROMless Mode



TL/DD/9765-7

FIGURE 2c. MICROWIRE/PLUS Timing

Pin Descriptions

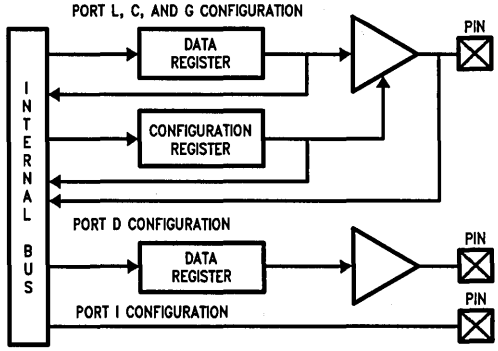
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

RESET is the master reset input. See Reset Description section.

The COP888CG contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CG memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CG. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/9765-8

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

- L0 MIWU
- L1 MIWU or CKX
- L2 MIWU or TDX
- L3 MIWU or RDX
- L4 MIWU or T2A
- L5 MIWU or T2B
- L6 MIWU or T3A
- L7 MIWU or T3B

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 serves as the dedicated CKO clock output. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLK Delay	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

- G0 INTR (External Interrupt Input)
- G2 T1B (Timer T1 Capture Input)
- G3 T1A (Timer T1 I/O)
- G4 SO (MICROWIRE™ Serial Data Output)
- G5 SK (MICROWIRE Serial Clock)
- G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

- G1 WDOUT WatchDog and/or Clock Monitor dedicated output
- G7 CKO Oscillator dedicated output or general purpose input

PORT I is an eight-bit Hi-Z input port. The 28- and 40-pin devices do not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the

Pin Descriptions (Continued)

accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

- I1 COMP1−IN (Comparator 1 Negative Input)
- I2 COMP1+IN (Comparator 1 Positive Input)
- I3 COMP1OUT (Comparator 1 Output)
- I4 COMP2−IN (Comparator 2 Negative Input)
- I5 COMP2+IN (Comparator 2 Positive Input)
- I6 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CG is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CG architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_C) cycle time.

There are six CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)
PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with RESET.

S is the 8-bit Data Segment Address Register used to extend the lower half of the address range (C0 to 7F) into 256 data segments of 128 bytes each.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CG consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CG vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE counter). Data memory is addressed directly by the instruction or indirectly by the B, X, SP pointers and S register.

The COP888CG has 192 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, B and C are memory mapped into this space at address locations 0FC to 0FF Hex respectively, with the other registers being available for general usage.

The instruction set of the COP888CG permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CG (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Data Memory Segment RAM Extension

Data memory address 0FF is used as a memory mapped location for the Data Segment Address Register (S) in the COP888CG. The upper bit of the Memory Address Register (MAR) is used to determine whether or not the segment register S is used to augment the RAM address. If the high order MAR bit is a 0, then the contents of the S register provide an additional 8 bits of RAM address. If the high order MAR bit is a 1, then the S register is not used, with the base address 080 to 0FF always being utilized. This organization allows a total of 256 data segments of 128 bytes each with an additional base segment of 128 bytes. Furthermore, all addressing modes are available for all segments.

The instructions that utilize the stack pointer (SP) always reference the stack as part of segment 0, regardless of the contents of the S register. The S register is not changed by these instructions. Consequently, the stack (used with subroutine linkage and interrupt) is always located in segment 0 (addresses 000 to 07F). The stack pointer will be initialized to location 06F after a RESET.

All special purpose registers (timers and autoreload/capture registers, control registers, UART registers, MICROWIRE shift register, Comparator Select Register, Multi-Input Wakeup control registers, WatchDog control register, etc.) as well as the B, X, and SP pointers and S register, are memory mapped into the base segment resident from memory locations 080 to 0FF. This base segment is selected whenever the high order MAR bit is a 1.

Data store memory is addressed directly by a single byte address within the instruction, or indirectly relative to the reference of the B or X pointers (each containing a single byte address). This single byte address allows an addressing range of 256 locations from 00 to FF (hex).

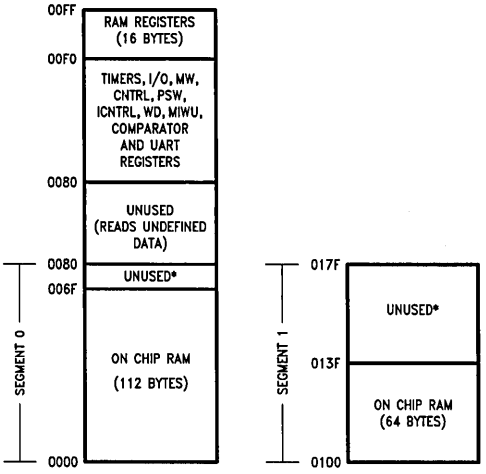
The S register is used to extend the lower half of the address range (00 to 7F hex) into 256 data segments of 128 bytes each. The S register must be charged under program control to move from one data segment (128 bytes) to another.

The additional 64 bytes of RAM (192 bytes total) in the COP888CG are memory mapped in the lower half of seg-

Pin Descriptions (Continued)

ment 1 (memory locations 100 to 13F hex). Segment 2 is reserved for E2 memory.

Figure 4 shows the COP888CG RAM organization.



TL/DD/9765-9

*Reads as all ones.

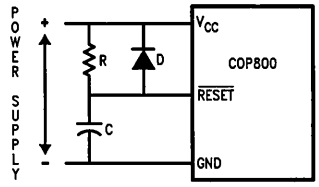
FIGURE 4. RAM Organization

Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for ports L, G and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is set high. The PC, PSW, ICNTRL, CNTRL, T2CNTRL and T3CNTRL control registers are cleared. The UART registers PSR, ENU (except that TBMT bit is set), ENUR and ENUI are cleared. The Comparator Select Register is cleared. The S register is initialized to zero. The Multi-Input Wakeup registers WKEN, WKEDG and WKPND are cleared. The stack pointer, SP, is initialized to 6F Hex.

The COP888CG comes out of RESET with both the WatchDog logic and the Clock Monitor detector armed, with the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor circuits are inhibited during RESET. The WatchDog service window bits being initialized high default to the maximum WatchDog service window of 64k t_C clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor error following RESET if the clock has not reached the minimum specified frequency at the termination of RESET. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16 t_C–32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 5* should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes. It is recommended that the components of the RC network be selected to provide a RESET delay of at least five times the power supply rise time or the minimum RESET pulse width, whichever is greater.



TL/DD/9765-10

RC > 5 × Power Supply Rise Time

FIGURE 5. Recommended RESET Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Figure 6 shows the Crystal and R/C diagrams.

EXTERNAL OSCILLATOR

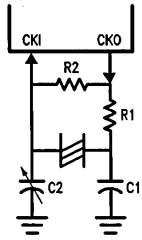
CKI can be driven by an external clock signal provided it meets the specified duty cycle, rise and fall times, and input levels.

CRYSTAL OSCILLATOR

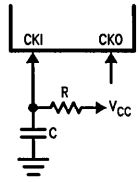
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.



TL/DD/9765-11



TL/DD/9765-12

FIGURE 6. Crystal and R/C Oscillator Diagrams

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. Comparator DC supply current when enabled—I6

Thus the total current drain, It, is given as

$$It = I1 + I2 + I3 + I4 + I5 + I6$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Some sample current drain values at V_{CC} = 5V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)	—	0.005

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE control register contains the following bits:

- SL1 & SL0 Select the MICROWIRE clock divide by (00 = 2, 01 = 4, 1× = 8)
- IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
- MSEL Selects G5 and G4 as MICROWIRE signals SK and SO respectively
- T1C0 Timer T1 Start/Stop control in timer modes 1 and 2
  Timer T1 Underflow Interrupt Pending Flag in timer mode 3
- T1C1 Timer T1 mode control bit
- T1C2 Timer T1 mode control bit
- T1C3 Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0

Bit 7 Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

- GIE Global interrupt enable (enables interrupts)
- EXEN Enable external interrupt
- BUSY MICROWIRE busy shifting flag
- EXPND External interrupt pending
- T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
- T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
- C Carry Flag
- HC Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE

Bit 7 Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

- T1ENB Timer T1 Interrupt Enable for T1B Input capture edge
- T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge
- μWEN Enable MICROWIRE/PLUS interrupt
- μWPND MICROWIRE/PLUS interrupt pending
- T0EN Timer T0 Interrupt Enable (Bit 12 toggle)
- T0PND Timer T0 Interrupt pending
- LPEN L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)
  Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PNDB	T1ENB

Bit 7 Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

- T2ENB Timer T2 Interrupt Enable for T2B Input capture edge
- T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge
- T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge
- T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)
- T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

Control Registers (Continued)

T2C1 Timer T2 mode control bit

T2C2 Timer T2 mode control bit

T2C3 Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB

Bit 7 Bit 0

T3CNTRL Register (Address X'00B6)

The T3CNTRL register contains the following bits:

T3ENB Timer T3 Interrupt Enable for T3B

T3PNDB Timer T3 Interrupt Pending Flag for T3B pin (T3B capture edge)

T3ENA Timer T3 Interrupt Enable for Timer Underflow or T3A pin

T3PNDA Timer T3 Interrupt Pending Flag (Autoload RA in mode 1, T3 Underflow in mode 2, T3a capture edge in mode 3)

T3C0 Timer T3 Start/Stop control in timer modes 1 and 2
Timer T3 Underflow Interrupt Pending Flag in timer mode 3

T3C1 Timer T3 mode control bit

T3C2 Timer T3 mode control bit

T3C3 Timer T3 mode control bit

T3C3	T3C2	T3C1	T3C0	T3PNDA	T3ENA	T3PNDB	T3ENB

Bit 7 Bit 0

Emulation and ROMless Modes

The COP888CG can address up to 32 kbytes of address space. If at power up, D2 is held at ground, the COP888CG executes from external memory. Port D is used to interface to external program memory. The address comes out in a serial fashion and the data from the external program memory is read back in a serial fashion. The Port D pins perform the following functions.

D0 Shifts in ROM data

D1 Shifts out lower eight bits of PC

D2 Places the μC in the ROMless mode if grounded at reset

D3 Shifts out upper eight bits of PC

D4 Data Shift Clock

D5 HALT Mask Option select pin
(D5 = 0) for HALT enable, D5 = 1 for HALT disable)

D6 Load Clock

D7 Shifts out recreated Port D data

The most significant bit of the data to come out on the D3 pin is a status signal.. It is used by the MOLE development system. This "lost" output port (D0–D7) can be accurately reconstructed with external components as shown in *Figure 6*, providing an accurate emulation.

The 44-pin and 40-pin versions of the COP888CG have a full complement of the D Port pins and can be used in the ROMless mode. However, it should be noted that the 44-pin device can only emulate itself and not the 40-pin or 28-pin devices as it has only 6 Port L pins while the other two devices have a full complement of Port L pins.

The 28-pin part cannot be used for emulation since it does not have the full complement of 8 D Port pins necessary for entering the ROMless mode.

Note that in the ROMless mode the D Port is recreated one full clock cycle behind the normal port timings.

Note: Standard parts used in the ROMless mode will operate only at a reduced frequency (to be defined).

The COP888CG device has a spare D pin (D5) in the emulation mode since only seven pins are required for emulation and recreation. This pin D5 is used in the emulation mode to enable or disable the HALT mask option feature.

Figure 7 shows the COP888CG Emulation Mode Schematic.

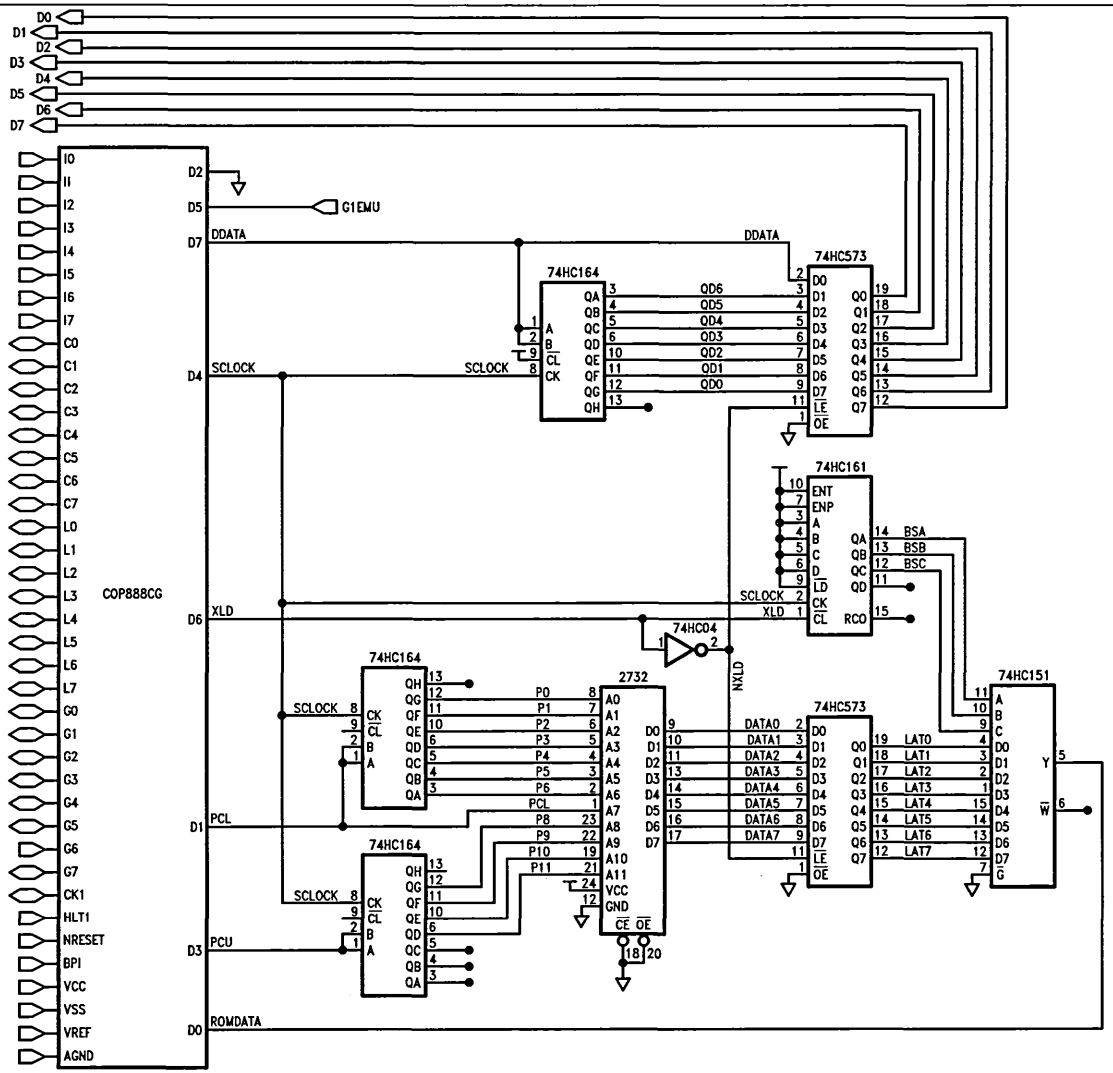


FIGURE 7. COP888CG Emulation Mode Schematic

Power Save Modes

The COP888CG offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CG is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock and timers, are stopped. The WatchDog logic on the COP888CG is disabled during the HALT mode. However, the clock monitor circuitry remains active. In the HALT mode, the power requirements of the COP888CG are minimal and the applied voltage (V_{CC}) may be decreased to V_r (V_r = 2.0V) without altering the state of the machine.

The COP888CG supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method preludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET input low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE time is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared at reset.

The COP88CG has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CG will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CG cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit remains active during HALT mode in order to ensure a clock monitor error if the COP888CG inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CG is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the Watch-Dog logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CG can be returned to normal operation with a RESET, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the twelfth bit (representing 4.096 ms at internal clock frequency of 1 MHz (t_c = 1 μs)) of the IDLE Timer toggles.

This toggle condition of the twelfth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

The user has the option of being interrupted with a transition on the twelfth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the COP888CG will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CG will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Timers

The COP888CG contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

TIMER T0 (IDLE TIMER)

The COP888CG supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_c. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)
WatchDog logic (See WatchDog description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the twelfth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency (t_c = 1 μs). A control flag T0EN allows the interrupt from the twelfth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

Timers (Continued)

TIMER T1, TIMER T2 AND TIMER T3

The COP888CG has a set of two powerful timer/counter blocks, T1, T2 and T3. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the three timer blocks, T1, T2 and T3 are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CG to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CG to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode.

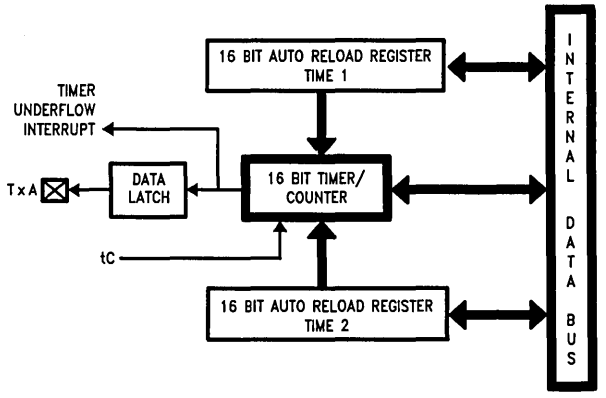
The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.



TL/DD/9765–14

FIGURE 8. Timer in PWM Mode

Timers (Continued)

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

Mode 3. Input Capture Mode

The COP888CG can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_C rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

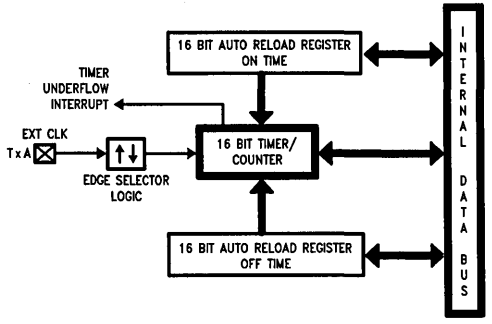
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

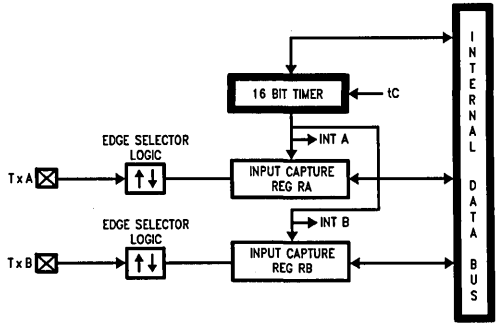
The timers T1, T2 and T3 have indentical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control



TL/DD/9765–15

FIGURE 9. Timer in External Event Counter Mode



TL/DD/9765–16

FIGURE 10. Timer in Input Capture Mode

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_C
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_C
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C

Detection of Illegal Conditions

The COP888CG will detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during RESET. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F (Segment 0), 140 to 17F (Segment 1), and all other segments (i.e., Segments 3 . . . etc.) is read as all 1's, which in turn will cause the program to return to address FFFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

a. Executing from undefined ROM

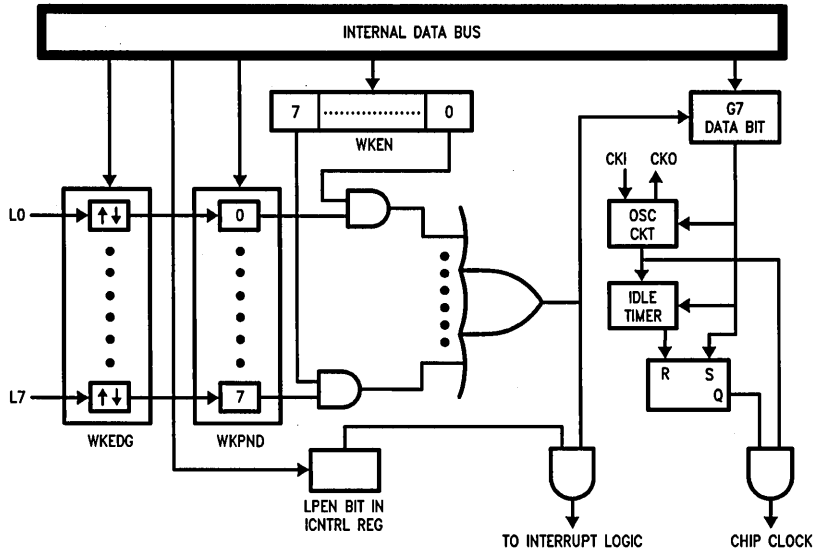
b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following RESET, but might not contain the same program initialization procedures).

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CG from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Multi-Input Wakeup (Continued)



TL/DD/9765–17

FIGURE 11. Multi-Input Wake Up Logic

Figure 11 shows the Multi-Input Wakeup logic for the COP888CG microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CG to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following RESET, since the L port inputs are left floating as a result of RESET.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called Reg: WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since the Reg: WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CG will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

All three registers Reg:WKEN, Reg:WKPND and Reg:WKEDG are read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG

Multi-Input Wakeup (Continued)

specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CG out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CG will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CG will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CG to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_C instruction cycle clock. The t_C clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during RESET, so the clock start up delay is not present following RESET with the RC clock options.

UART

The COP888CG contains a full-duplex software programmable UART. The UART *(Figure 12)* consists of a transmit shift register, a receiver shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR), a UART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framming, data overrun and parity errors while the UART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the UART's attention mode of operation and providing additional receiver/transmitter status information via RCVG and XMTG bits. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts. A control flag in this register can also select the UART mode of operation: asynchronous or synchronous.

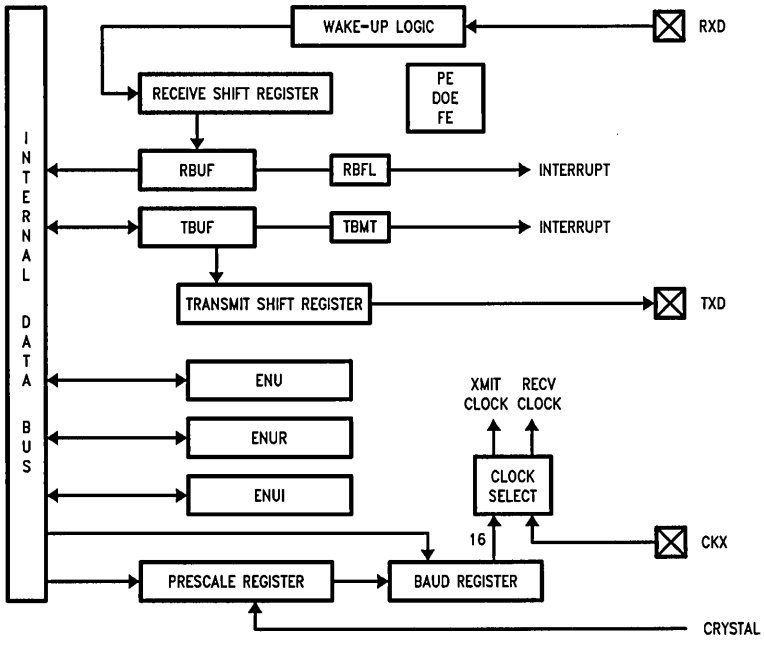


FIGURE 12. UART Block Diagram

UART (Continued)

UART CONTROL AND STATUS REGISTERS

The operation of the UART is programmed through three registers: ENU, ENUR and ENUI. The function of the individual bits in these registers is as follows:

Control and Status Register (Byte at 0BA)

PEN	PSEL1	XBIT9/ PSEL0	CHL1	CHL0	ERR	RBFL	TBMT
0RW	0RW	0RW	0RW	0RW	0R	0R	1R

Bit 7 Bit 0

ENUR-UART Receive Control and Status Register (Byte at 0BB)

DOE	FE	PE	SPARE	RBIT9	ATTN	XMTG	RCVG
0RD	0RD	0RD	0RW*	0R	0RW	0R	0R

Bit7 Bit0

ENUI-UART Interrupt and Clock Source Register (Byte at 0BC)

STP2	STP78	ETDX	SSEL	XRCLK	XTCLK	ERI	ETI
0RW	0RW	0RW	0RW	0RW	0RW	0RW	0RW

Bit7 Bit0

*Bit is not used.

0 Bit is cleared on reset.

1 Bit is set to one on reset.

R Bit is read-only; it cannot be written by software.

RW Bit is read/write.

D Bit is cleared on read; when read by software as a one, it is cleared automatically. Writing to the bit does not affect its state.

Associated I/O Pins

Data is transmitted on the TDX pin and received on the RDX pin. TDX is the alternate function assigned to Port L pin L2; it is selected by setting ETDX (in the ENUR register) to one. RDX is an inherent function of Port L pin L3, requiring no setup.

The baud rate clock for the UART can be generated on-chip, or can be taken from an external source. Port L pin L1 (CKX) is the external clock I/O pin. The CKX pin can be either an input or an output, as determined by Port L Configuration and Data registers (Bit 1). As an input, it accepts a clock signal which may be selected to drive the transmitter and/or receiver. As an output, it presents the internal Baud Rate Generator output.

UART OPERATION

The UART has two modes of operation: asynchronous mode and synchronous mode.

ASYNCHRONOUS MODE

This mode is selected by resetting the SSEL (in the ENUI register) bit to zero. The input frequency to the UART is 16 times the baud rate.

The TSFT and TBUF registers double-buffer data for transmission. While TSFT is shifting out the current character on the TDX pin, the TBUF register may be loaded by software with the next byte to be transmitted. When TSFT finishes transmitting the current character the contents of TBUF are transferred to the TSFT register and the Transmit Buffer Empty Flag (TBMT in the ENU register) is set. The TBMT flag is automatically reset by the UART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the UART is transmitting. This bit gets reset at the end of the last frame (end of last Stop bit). TBUF is a read/write register.

The RSFT and RBUF registers double-buffer data being received. The UART receiver continually monitors the signal on the RDX pin for a low level to detect the beginning of a Start bit. Upon sensing this low level, it waits for half a bit time and samples again. If the RDX pin is still low, the receiver considers this to be a valid Start bit, and the remaining bits in the character frame are each sampled a single time, at the mid-bit position. Serial data input on the RDX pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the RBUF register and the Received Buffer Full Flag (RBFL) is set. RBFL is automatically reset when software reads the character from the RBUF register. RBUF is a read only register. There is also the RCVG bit which is set high when a framing error occurs and goes low once RDX goes high. TBMT, XMTG, RBFL and RCVG are read only bits.

SYNCHRONOUS MODE

In this mode data is transferred synchronously with the clock. Data is transmitted on the rising edge and received on the falling edge of the synchronous clock.

This mode is selected by setting SSEL bit in the ENUI register. The input frequency to the UART is the same as the baud rate.

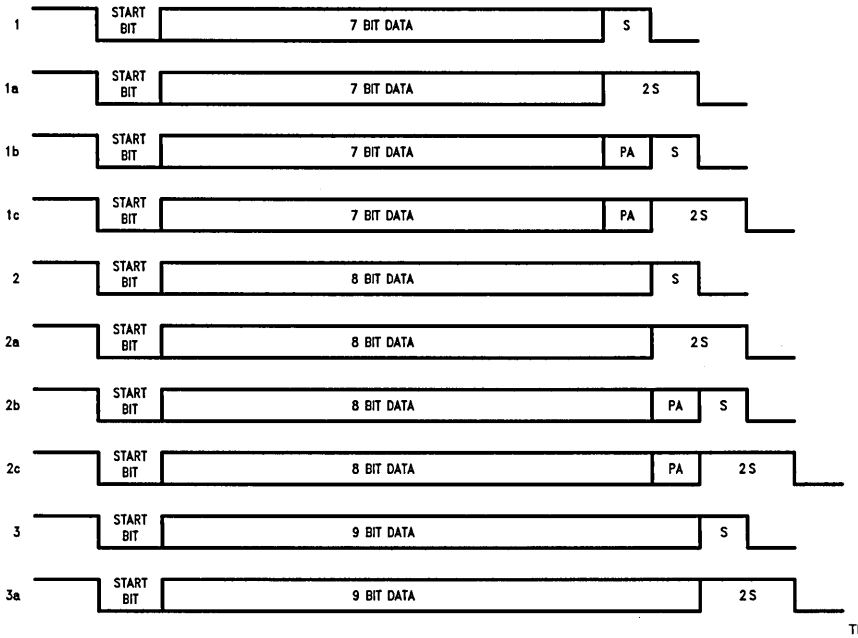
When an external clock input is selected at the CKX pin, data transmit and receive are performed synchronously with this clock through TDX/RDX pins.

If data transmit and receive are selected with the CKX pin as clock output, the μC generates the synchronous clock output at the CKX pin. The internal baud rate generator is used to produce the synchronous clock. Data transmit and receive are performed synchronously with this clock.

FRAMING FORMATS

The UART supports several serial framing formats *(Figure 13)*. The format is selected using control bits in the ENU register.

UART OPERATION (Continued)



TL/DD/9765-19

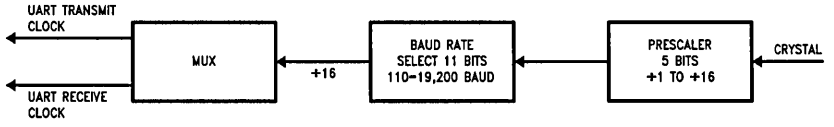
FIGURE 13. Framing Formats

Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the UART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a source selected in the PSR and BAD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1–16 (increments of 0.5) prescaler and an 11-bit binary counter. *(Figure 14)* The divide factors are specified through two read/write registers shown in *Figure 15*. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

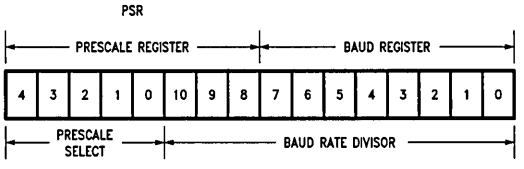
As shown in Table I, a Prescale Factor of 0 corresponds to NO CLOCK. NO CLOCK condition is the UART power down mode where the UART clock is turned off for power saving purpose. The user must also turn the UART clock off when a different baud rate is chosen.

The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in Table I. Therer are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a x16 clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (Table II). Other baud rates may be created by using appropriate divisors. The x16 clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.



TL/DD/9765-20

FIGURE 14. UART BAUD Clock Generation



TL/DD/9765-21

FIGURE 15. UART BAUD Clock Divisor Registers

Baud Clock Generation (Continued)

TABLE I. Prescaler Factors

Prescaler Select	Prescaler Factor
00000	NO CLOCK
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11001	13
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

TABLE II. Baud Rate Divisors (1.8432 MHz Prescaler Output)

Baud Rate	Baud Rate Divisor (N-1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5
38400	2

The entries in Table II assume a prescaler output of 1.8432 MHz.

As an example, considering the Asynchronous Mode and a CKI clock of 4.608 MHz, the prescaler factor selected is:

$$4.608/1.8432 = 2.5$$

The 2.5 entry is available in Table I. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (Table II) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in Table II is 5.

$N - 1 = 5$ ($N - 1$ is the contents of Baud Rate Divisor)

$N = 6$ (N is the divisor)

Baud Rate = 1.8432 MHz $(16 \times 6) = 19200$

The divide by 16 is performed because in the asynchronous mode, the input frequency to the UART is 16 times the baud rate. The equation to calculate baud rates is given below.

The actual Baud Rate may be found from:

$$BR = Fc/(16 \times N \times P)$$

Where:

BR is the Baud Rate

Fc is the CKI frequency

N is one plus the value of the Baud Rate Divisor (Table III).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (Table I)

Note: In the Synchronous Mode, the divisor 16 is replaced by one.

Example:

Asynchronous Mode:

Crystal Frequency = 5 MHz

Desired baud rate = 9600

Using the above equation $N \times P$ can be calculated first.

$$N \times P = (5 \times 10^6)/(16 \times 9600)$$

Now 32.552 is divided by each Prescaler Factor (Table II) to obtain a value closest to an integer. This factor happens to be 6.5 ($P = 6.5$).

$$N = 32.552/6.5 = 5.008 \ (N = 5)$$

The Baud Rate Divisor value should be 4 ($N - 1$).

Using the above values calculated for N and P:

$$BR = (5 \times 10^6)/(16 \times 5 \times 6.5) = 9615.384$$

$$\% \text{ error} = (9615.385 - 9600)/9600 = 0.16$$

Attention Mode

The UART Receiver section supports an alternate mode of operation, referred to as ATTENTION Mode. This mode of operation is selected by the ATTN bit in the ENUR register. The data format for transmission must also be selected as having nine Data bits and either 7/8, one or two Stop bits.

The ATTENTION mode of operation is intended for use in networking the COP888CG with other processors. Typically in such environments the messages consists of device addresses, indicating which of several destinations should receive them, and the actual data. This Mode supports a scheme in which addresses are flagged by having the ninth bit of the data field set to a 1. If the ninth bit is reset to a zero the byte is a Data byte.

While in ATTENTION mode, the UART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the UART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if UART

Attention Mode (Continued)

Receiver interrupts are enabled. The ATTN bit is also cleared automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

Operation of the UART Transmitter is not affected by selection of this Mode. The value of the ninth bit to be transmitted is programmed by setting XBIT9 appropriately. The value of the ninth bit received is obtained by reading RBIT9. Since this bit is located in ENUR register where the error flags reside, a bit operation on it will reset the error flags.

Comparators

The COP888CG has two differential comparators. Ports I1–I3 and I4–I6 are used for the comparators. The output of the comparators are brought out to the pins. The following is the Port I assignment.

- I1 Comparator1 negative input
- I2 Comparator1 positive input
- I3 Comparator1 output
- I4 Comparator2 negative input
- I5 Comparator2 positive input
- I6 Comparator2 output

COMPARATOR SELECT REGISTER (ADDRESS X'00B7)

The register contains the following bits:

- CMP1EN Enables comparator1 ("1" = enable)
- CMP1RD Reads comparator1 output internally (CMP1EN = 1, CMP1OE = 0)
- CMP1OE Enables comparator1 output to pin 13 ("1" = enable), CMP1EN bit must be set to enable this function.
- CMP2EN Enables comparator2 ("1" = enable)
- CMP2RD Reads comparator2 output internally (CMP2EN = 1, CMP2OE = 0)
- CMP2OE Enables comparator2 output to pin I6 ("1" = enable), CMP2EN bit must be set to enable this function.

Unused	CMP 2OE	CMP 2RD	CMP 2EN	CMP 1OE	CMP 1RD	CMP 1EN	Unused
Bit 7							Bit 0

The Comparator Select Register is cleared on RESET (the comparators are disabled). To save power the program should also disable the comparators before the μC enters the HALT/IDLE modes.

Comparator outputs have the save spec as Ports L and G except that the rise and fall times are symmetrical.

Interrupts

The COP888CG supports a vectored interrupt scheme. It supports a total of fourteen interrupt sources. The following table lists all the possible COP888CG interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_C cycles to execute.

Interrupts (Continued)

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	T0 Bit 12 Toggle	0yF8–0yF9
(4)	Timer T1	T1 Underflow/ T1A Capture Edge	0yF6–0yF7
(5)	Timer T1	T1B Capture Edge	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
	UART	Receive	0yEE–0yEF
	UART	Transmit	0yEC–0yED
(7)	Timer T2	T2 Underflow/ T2A Capture Edge	0yEA–0yEB
(8)	Timer T2	T2B Capture Edge	0yE8–0yE9
	Reserved	for Future Use	0yE6–0yE7
	Reserved	for Future Use	0yE4–0yE5
(9)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(10) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 16 shows the COP888CG Interrupt block diagram.

Interrupts (Continued)

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to RESET, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit **(not accessible by the user)** is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction.

It is cleared by RESET and by the RPND instruction.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

The COP888CG contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

WatchDog

The COP888CG WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDCNT which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table III shows the WDCNT register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 ad 6 of the WDCNT register allow the user to pick an upper limit of the service window.

Table IV shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDCNT register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDCNT Register is the Clock Monitor Select bit.

TABLE III

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE IV

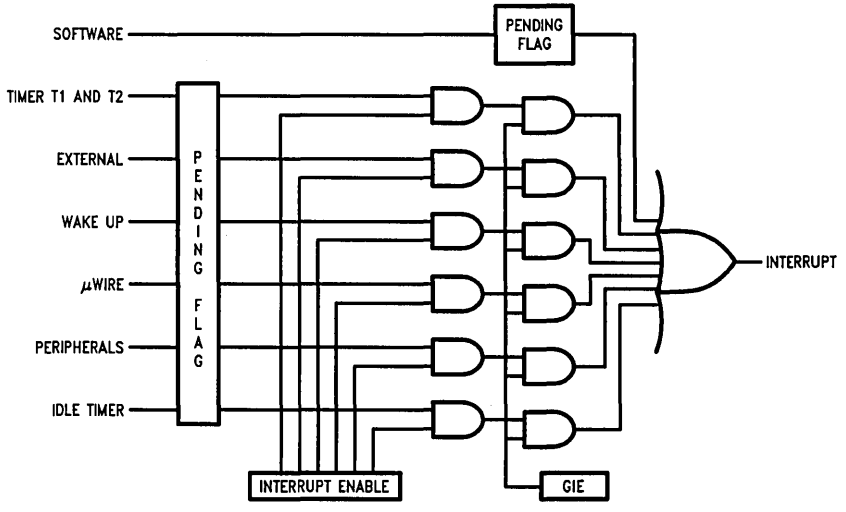
WDCNT Bit 7	WDCNT Bit 6	Service Window (Lower-Upper Limits)
0	0	2k–8k t_C Cycles
0	1	2k–16k t_C Cycles
1	0	2k–32k t_C Cycles
1	1	2k–64k t_C Cycles

Clock Monitor

The Clock Monitor aboard the COP888CG can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_C$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WatchDog Operation

The WatchDog and Clock Monitor are disabled during $\overline{\text{RESET}}$. The COP888CG comes out of $\overline{\text{RESET}}$ with the WatchDog armed, the WatchDog Window Select bits (bits 6,



TL/DD/9765–22

FIGURE 16. COP888CG Interrupt Block Diagram

WatchDog Operation (Continued)

7 of the WDCNT Register) set, and the Clock Monitor bit (bit 0 of the WDCNT Register) enabled. Thus, a Clock Monitor error will occur after coming out of RESET, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDCNT register can be written to only once after RESET and the key data (bits 5 through 1 of the WDCNT Register) must match to be a valid write. This write to the WDCNT register involves two irrevocable choices: (i) the selection of the WatchDog service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDCNT Register involves selecting or deselecting the Clock Monitor, select the WatchDog service window and match the WatchDog key data. Subsequent writes to the WDCNT register will compare the value being written by the user to the WatchDog service window value and the key data (bits 7 through 1) in the WDCNT Register. Table V shows the sequence of events that can occur.

The user must service the WatchDog at least once before the upper limit of the serivce window expires. The WatchDog may not be serviced more than once in every lower limit of the service window. The user may service the WatchDog as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDCNT Register is also counted as a WatchDog service.

The WatchDog has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WatchDog, the logic will pull the WDOUT (G1) pin low for an additional 16 t_C–32 t_C cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CG will stop forcing the WDOUT output low.

The WatchDog service window will restart when the WDOUT pin goes inactive. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WatchDog service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed at RESET, but if it powers up low then the WatchDog will time out and disable.

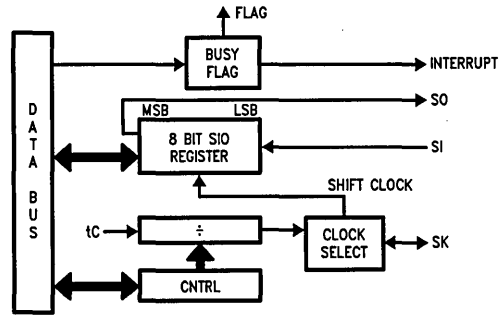
The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_C–32 t_C clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_C > 10$ kHz—No clock rejection.

$1/t_C < 10$ Hz—Guaranteed clock rejection.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CG to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E²PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 13* shows a block diagram of the MICROWIRE logic.



TL/DD/9765–23

FIGURE 17. MICROWIRE Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE mode. To use the MICROWIRE, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table IV details the different clock rates that may be selected.

TABLE V

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WatchDog Output
Mismatch	Don't Care	Don't Care	Error: Generate WatchDog Output
Don't Care	Don't Care	Mismatch	Error: Generate WatchDog Output

TABLE VI

SL1	SL0	SK
0	0	$2 \times t_C$
0	1	$4 \times t_C$
1	x	$8 \times t_C$

Where t_C is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP888CG may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 14* shows how two COP888CG microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock forthe SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CG. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table VII summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SKclock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table VII summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

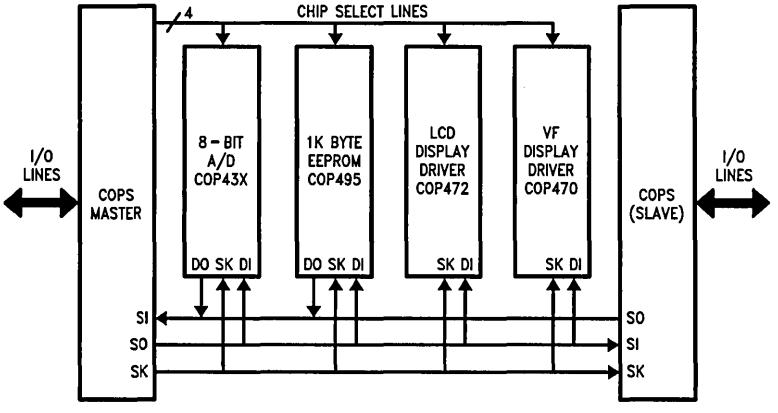
The COP888CG allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock ahd shifted out on the rising edge of the SK clock. In the alternate SK phase mode the SIO register is shifted on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE VII

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE Master
0	1	TRI-STATE	Int. SK	MICROWIRE Master
1	0	SO	Ext. SK	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE Slave



TL/DD/9765-24

FIGURE 18. MICROWIRE Application

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address S/MAR	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads As All Ones)
xx80 to xxAF	Unused RAM Address Space (Reads Undefined Data)
xxB0	Timer T3 Lower Byte
XXB1	Timer T3 Upper Byte
xxB2	Timer T3 Autoload Register T3RA Lower Byte
xxB3	Timer T3 Autoload Register T3RA Upper Byte
xxB4	Timer T3 Autoload Register T3RB Lower Byte
xxB5	Timer T3 Autoload Register T3RB Upper Byte
xxB6	Timer T3 Control Register
xxB7	Comparator Select Register
xxB8	UART Transmit Buffer (TBUF)
xxB9	UART Receive Buffer (RBUF)
xxBA	UART Control and Status Register (ENU)
xxBB	UART Receive Control and Status Register (ENUR)
xxBC	UART Interrupt and Clock Source Register (ENUI)
xxBD	UART Baud Register (BAUD)
xxBE	UART Prescale Select Register (PSR)
xxBF	Reserved for UART
xxC0	Timer T2 Lower Byte
xxC1	Timer T2 Upper Byte
xxC2	Timer T2 Autoload Register T2RA Lower Byte
xxC3	Timer T2 Autoload Register T2RA Upper Byte
xxC4	Timer T2 Autoload Register T2RB Lower Byte
xxC5	Timer T2 Autoload Register T2RB Upper Byte
xxC6	Timer T2 Control Register
xxC7	WatchDog Service Register (Reg:WDCNT)
xxC8	MIWU Edge Select Register (Reg:WKEDG)
xxC9	MIWU Enable Register (Reg:WKEN)
xxCA	MIWU Pending Register (Reg:WKPND)
xxCB	Reserved
xxCC	Reserved
xxCD to xxCF	Reserved

Address S/MAR	Contents
xxD0	Port L Data Register
xxD1	Port L Configuration Register
xxD2	Port L Input Pins (Read Only)
xxD3	Reserved for Port L
xxD4	Port G Data Register
xxD5	Port G Configuration Register
xxD6	Port G Input Pins (Read Only)
xxD7	Port I Input Pins (Read Only)
xxD8	Port C Data Register
xxD9	Port C Configuration Register
xxDA	Port C Input Pins (Read Only)
xxDB	Reserved for Port C
xxDC	Port D
xxDD to DF	Reserved for Port D
xxE0 to xxE5	Reserved for EE Control Registers
xxE6	Timer T1 Autoload Register T1RB Lower Byte
xxE7	Timer T1 Autoload Register T1RB Upper Byte
xxE8	ICNTRL Register
xxE9	MICROWIRE Shift Register
xxEA	Timer T1 Lower Byte
xxEB	Timer T1 Upper Byte
xxEC	Timer T1 Autoload Register T1RA Lower Byte
xxED	Timer T1 Autoload Register T1RA Upper Byte
xxEE	CNTRL Control Register
xxEF	PSW Register
xxF0 to FB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	Reserved
0100–013F	On-Chip RAM Bytes (64 bytes)

Reading memory locations 0070H–007FH (Segment 0) will return all ones. Reading unused memory locations 0080H–00AFH (Segment 0) will return undefined data. Reading unused memory locations 0140–017F (Segment 1) will return all ones. Reading memory locations from other Segments (i.e., Segment 2, Segment 3, ... etc.) will return all ones.

Addressing Modes

The COP888CG has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CG. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the LBI instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
<−	Loaded with
<−>	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A <− A + Meml
ADC	A,Meml	ADD with Carry	A <− A + Meml + C, C <− Carry HC <− Half Carry
SUBC	A,Meml	Subtract with Carry	A <− A − Meml + C, C <− Carry HC <− Half Carry
AND	A,Meml	Logical AND	A <− A and Meml
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A <− A or Meml
XOR	A,Meml	Logical EXclusive OR	A <− A xor Meml
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF EQual	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A not = Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B not = Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg <− Reg− 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A <−> Mem
LD	A,Meml	LoaD A with Memory	A <− Meml
LD	B,Imm	LoaD B with Immed.	B <− Imm
LD	Mem,Imm	LoaD Memory Immed	Mem <− Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg <− Imm
X	A, [B ±]	EXchange A with Memory [B]	A <−> [B], (B <− B ±1)
X	A, [X ±]	EXchange A with Memory [X]	A <−> [X], (X <− ±1)
LD	A, [B±]	LoaD A with Memory [B]	A <−[B], (B <−B ±1)
LD	A, [X±]	LoaD A with Memory [X]	A <− [X], (X <− X±1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] <−Imm, (B <− B±1)
CLR	A	CLeaR A	A <− 0
INC	A	INCrement A	A<− A + 1
DEC	A	DECrementA	A <− A − 1
LAID		Load A InDirect from ROM	A <− ROM (PU,A)
DCOR	A	Decimal CORrect A	A <− BCD correction (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C −> A7 −> ... −> A0 −> C
RLC	A	Rotate A Left thru C	C <− A7 <− ... <− A0 <− C
SWAP	A	SWAP nibbles of A	A7 ... A4 <−> A3 ... A0
SC		Set C	C <− 1, HC <− 1
RC		Reset C	C <− 0, HC <− 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP <− SP + 1, A <− [SP]
PUSH	A	PUSH A onto the stack	[SP] <− A, SP <− SP − 1
VIS		Vector to Interrupt Service Routine	PU <− [VU], PL <− [VL]
JMPL	Addr.	Jump absolute Long	PC <− ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9 ... 0 <− i (i = 12 bits)
JP	Disp.	Jump relative short	PC <− PC + r (r is −31 to +32, not 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] <− PL, [SP−1] <− PU,SP−2, PC <− ii
JSR	Addr	Jump SubRoutine	[SP] <− PL, [SP−1] <− PU,SP−2, PC9 ... 0 <− i
JID		Jump InDirect	PL <− ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL <−[SP], PU <− [SP−1]
RETSK		RETurn and SKip	SP + 2, PL <−[SP],PU <−[SP−1],Skip <− 1
RETI		RETurn from Interrupt	SP + 2, PL <−[SP],PU <−[SP−1],GIE<− 1
INTR		Generate an Interrupt	[SP] <− PL, [SP−1] <− PU, SP−2, PC <− 0FF
NOP		No OPeration	PC <− PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time (1 μs at 10 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 10 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B, Imm				1/1			(IF B < 16)
LD B, Imm				2/2			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

* = > Memory location addressed by B or X or directly.

COP888CG Opcode Table

Upper Nibble Along X-Axis
Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP −15	JP −31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	0
JP −14	JP −30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP −13	JP −29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	2
JP −12	JP −28	LD 0F3, # i	DRSZ 0F3	X A, [X−]	X A,[B−]	IFGT A,#i	IFGT A,[B]	3
JP −11	JP −27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	4
JP −10	JP −26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	5
JP −9	JP −25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	6
JP −8	JP −24	LD 0F7, # i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	7
JP −7	JP −23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	8
JP −6	JP −22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	9
JP −5	JP −21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+],#i	INCA	A
JP −4	JP −20	LD 0FB, # i	DRSZ 0FB	LD A,[X−]	LD A,[B−]	LD [B−],#i	DECA	B
JP −3	JP −19	LD 0FC, # i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	C
JP −2	JP −18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP −1	JP −17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B],#i	RET	E
JP −0	JP −16	LD 0FF, # i	DRSZ 0FF	*	*	LD B,#i	RETI	F

COP888CG Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B,#0F	IFBNE 0	JSR x000–x0FF	JMP x000–x0FF	JP +17	INTR	0
IFBIT 1,[B]	*	LD B,#0E	IFBNE 1	JSR x100–x1FF	JMP x100–x1FF	JP +18	JP + 2	1
IFBIT 2,[B]	*	LD B,#0D	IFBNE 2	JSR x200–x2FF	JMP x200–x2FF	JP +19	JP + 3	2
IFBIT 3,[B]	*	LD B,#0C	IFBNE 3	JSR x300–x3FF	JMP x300–x3FF	JP +20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400–x4FF	JMP x400–x4FF	JP +21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500–x5FF	JMP x500–x5FF	JP +22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600–x6FF	JMP x600–x6FF	JP +23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700–x7FF	JMP x700–x7FF	JP +24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800–x8FF	JMP x800–x8FF	JP +25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900–x9FF	JMP x900–x9FF	JP +26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00–xAFF	JMP xA00–xAFF	JP +27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00–xBFF	JMP xB00–xBFF	JP +28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00–xCFF	JMP xC00–xCFF	JP +29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00–xDFF	JMP xD00–xDFF	JP +30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00–xEFF	JMP xE00–xEFF	JP +31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00–xFFF	JMP xF00–xFFF	JP +32	JP + 16	F

Where,

i is the immediate data

Md is a directly addressed memory location

* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CG mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

```
OPTION 1: CLOCK CONFIGURATION
  = 1     Crystal Oscillator (CKI/10)
              G7 (CKO) is clock generator
              output to crystal/resonator
              CKI is the clock input
  = 2     Single-pin RC controlled
              oscillator (CKI/10)
              G7 is available as a HALT
              restart and/or general purpose
              input

OPTION 2: HALT
  = 1     Enable HALT mode
  = 2     Disable HALT mode

OPTION 3: COP888CG BONDING
  = 1     44-Pin PCC
  = 2     40-Pin DIP
  = 3     28-Pin PCC
  = 4     28-Pin DIP
```

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	TBD	Programmer's Manual		TBD

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

Information System

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud modem, and a telephone.

Voice: (408) 721-5582

Modem: (408) 739-1162

Baud: 300 or 1200 Baud

Set-up: Length: 8-Bit
Parity: None
Stop Bit:

Operation: 24 Hours, 7 Days

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

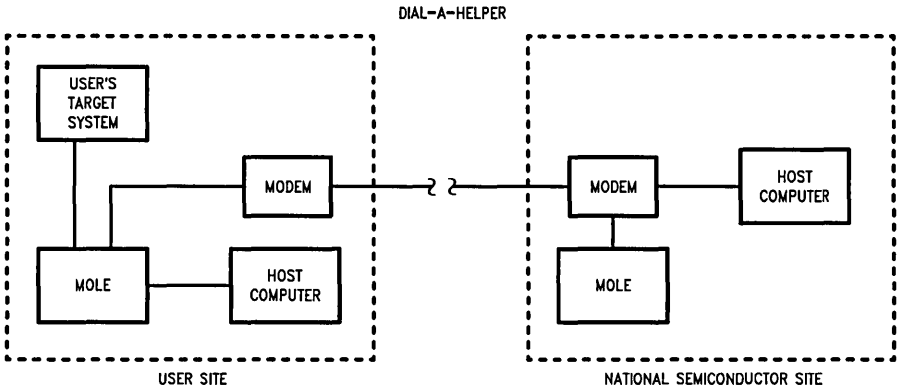
Order P/N: MOLE-DIAL-A-HLP
Information System Package Contents
Dial-A-Helper User Manual P/N
Public Domain Communications Software

Factory Applications Support

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occuring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will responed to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.



TL/DD/9765-25

Section 3

COPS Applications

Section 3 Contents

Easy Logarithms for COP400

National Semiconductor
COP Brief 2

Logarithms have long been a convenient tool for the simplification of multiplication, division, and root extraction. Many assembly language programmers avoid the use of logarithms because of supposed complexity in their application to binary computers. Logarithms conjure up visions of time consuming iterations during the solution of a long series. The problem is far simpler than imagined and its solution yields, for the applications programmer, the classical benefits of logarithms:

1) Multiplication can be performed by a single addition.

2) Division can be performed by a single subtraction.

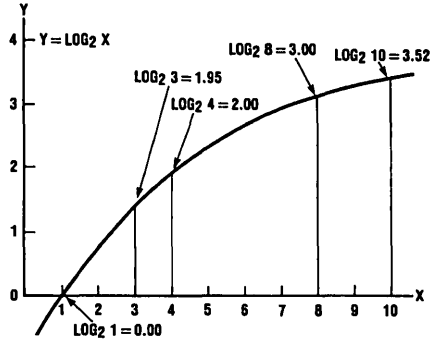
3) Raising a number to a power involves a single multiply.

4) Extracting a root involves a single divide.

When applied to binary computer operation logarithms yield two further important advantages. First, a broad range of values can be handled without resorting to floating point techniques (other than implied by the characteristic). Second, it is possible to establish the significance of an answer during the body of a calculation, again, without resorting to floating point techniques.

Implementation of $base_{10}$ logarithms in a binary system is cumbersome and unnecessary since logarithmic functions can be implemented in a number system of any base. The techniques presented here deal only with logarithms to the $base_2$.

A logarithm consists of two parts: an integer characteristic and a fractional mantissa.



TL/DD/6942–1

	CHARACTERISTIC	MANTISSA
LOG_2 3 =	1	0.95
LOG_2 4 =	2	0.00
LOG_2 8 =	3	0.00
LOG_2 10 =	3	0.52

FIGURE 1. The Logarithmic Function and Some Example Values

In *Figure 1* some points on the logarithmic curve are identified and evaluated to the $base_2$. Notice that the characteristic in each case represents the highest even power of 2 contained in the value of X. This is readily seen when binary notation is used.

X_{10}	X_2					Log_2 X Characteristic	Log_2 X Where X = Even Power of 2
	2^4	2^3	2^2	2^1	2^0		
3	0	0	0	1▲	1	1	
4	0	0	1▲	0	0	2	010.0000
8	0	1▲	0	0	0	3	011.0000
10	0	1▲	0	1	0	3	

FIGURE 2. Identification of the Characteristic

In *Figure 2* each point evaluated in *Figure 1* has been repeated using binary notation. An arrow subscript indicates the highest even power of 2 appearing in each value of X. Notice that in X = 3 the highest even power of 2 is 2^1. Thus the characteristic of the $\log_2$ 3 is 1. Where X = 10 the characteristic of the $\log_2$ 10 is 3.

To find the $\log_2$ X is very easy where X is an even power of 2. We simply shift the value of X left until a carry bit emerges from the high order position of the register. This procedure is illustrated in *Figure 3*. This characteristic is found by counting the number of shifts required and subtracting the result from the number of bits in the register. In practice it is easier to being with the number of bits and count down once prior to each shift.

Counter for Characteristic	Value of X in Binary		
1000	0000	1000	Initial
0111	0001	0000	First Shift
0110	0010	0000	Second Shift
0101	0100	0000	Third Shift
0100	1000	0000	Fourth Shift
0011	0000	0000	Fifth Shift
Characteristic	**Mantissa**		**Final**
011.0000	0000		Log_2 X = 3.00

FIGURE 3. Conversion to $Base_2$ Logarithm by Base Shift

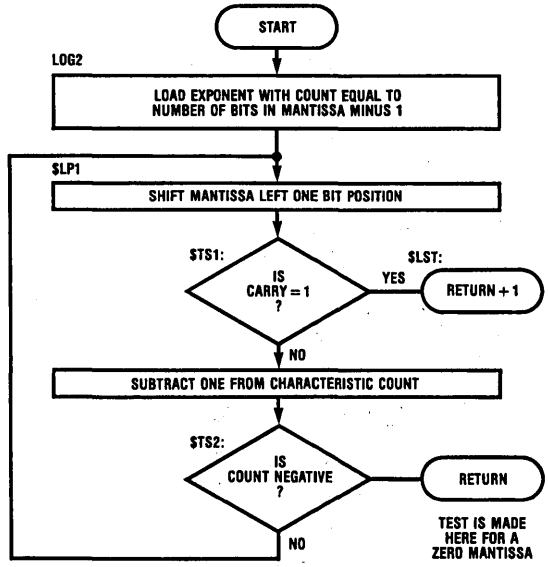
Examination of the final value obtained in *Figure 3* reveals no bits in the mantissa. The value 3 in the characteristic, however, indicates that a bit did exist in the 2^3 position of the original number and would have to be restored in order to reconstruct the original value (antilog).

The log of any even power of 2 can be found in this way:

Decimal	Binary	Log_2
128	10000000	0111.00000000
64	01000000	0110.00000000
32	00100000	0101.00000000
4	00000100	0010.00000000
2	00000010	0001.00000000
1	00000001	0000.00000000

FIGURE 4. Base$_2$ Logarithms of Even Powers of 2

A simple flow chart, and program, can be devised for generating the values found in the table and, as will be apparent, a straight line approximation for values that are not even powers of 2. The method, as already illustrated in *Figure 3,* involves only shifting a binary number left until the most significant bit moves into the carry position. The characteristic is formed by counting. Since a carry on each successive shift will yield a decreasing power of 2, we must start the characteristic count with the number of bits in the binary value (x) and count down one each shift.



TL/DD/6942-2

FIGURE 5. Log Flowchart

```
COP CROSS ASSEMBLER     PAGE:  1
LOGS

 1                          ; TITLE LOGS                ; BINARY LOGARITHMS
 2
 3          01A4                        . CHIP 420
 4
 5                          ; ·····→ CONVERT TO LOGARITHM ←·····;
 6                          ;
 7                          ;                          RAM ASSIGNMENT
 8                          ;
 9                          ; DIGIT:    15  14  13  12  11  10  09  08  07  06  05  04  03  02  01  00
10                          ;
11                          ; REG 0                                     CH  HM  LM                  TEMP
12                          ;
13                          ; REG 1                         CH  HM  LM                              TEMP
14                          ;
15                          ; REG 2                             TEMP                            CH  HM  LM
16                          ;
17                          ; REG 3                                             TEMP            CH  HM  LM
18                          ;
19                          ;
20                                      . LOCAL
21                          ;
22                          ; CH, HM, LM REPRESENT ANY THREE SEQUENTIAL MEMORY DIGITS. THEY
23                          ; MAY BE DEFINED IN ANY REGISTER. THE SYMBOLIC NOTATION CH, HM,
24                          ; AND LM ARE USED FOR ADDRESSING TO ALLOW USER FLEXIBILITY.
25                          ; UPON ENTRY TO THE ROUTINE HM AND LM CONTAIN THE HI AND LO
26                          ; OF SOME VALUE X. THE MEMORY POINTER MUST CONTAIN THE ADDRESS
27                          ; OF THE CHARACTERISTIC (CH). THE CONTENTS OF THIS LOCATION ARE
28                          ; IGNORED AND ARE LOST DURING EXECUTION.
29                          ;
30                          ; UPON EXIT CH, HM, LM CONTAIN A STRAIGHT LINE APPROXIMATION OF
31                          ; THE LOG BASE 2 OF X. CH = CHARACTERISTIC  HM = HI ORDER MANTISSA
32                          ; LM = LO ORDER MANTISSA. AN 8 BIT MEMORY AREA (TEMP) IS USED IN
33                          ; THE REGISTER OPPOSITE DURING THE CORRECTION OF A STRAIGHT
34                          LINE APPROXIMATION OF A LOG OR AN ANTILOG.
35                          ;               A TEST IS MADE FOR X = 0. IF THE VALUE OF X
36                          ; IS NOT ZERO AN INSTRUCTION IS SKIPPED UPON RETURN
37                          ; TO THE CALLING ROUTINE.
38                          ;
39                          ;                                                  — EXAMPLE —
40                          ;
41                          ; SUBROUTINE CALL                                      JSR   LOG2
42                          ; RETURN HERE IF X = 0 ⟶                               JP    ZERO
43                          ; RETURN HERE IF X>0 ⟶                                 CONTINUE
44                          ;
45                          ;
46                          ;
47                          ;
48                          ;
49                          ;
50     000     00           LOG2:       CLRA
51     001     57                      AISC        07                  ; TO REG LENGTH – 1.
52     002     06                      X                               ; STORE IN MEMORY.

COP CROSS ASSEMBLER     PAGE:  2
LOGS

53     003     A4           $LP1:       JSRP        SDB2                ; SET ADDRESS POINTER
54                                                                      ; BACK 2 DIGITS.
55     004     A9                      JSRP        SHLR                ; RESET CARRY AND SHIFT
56                                                                      ; REG LEFT ONE BIT.
57     005     20           $TS1:       SKC                             ; IS CARRY = 1 YET?
58     006     C8                      JP          $NO                 ; NO — KEEP GOING.
59     007     49           $LST:       RETSK                           ; YES — FINISHED!!
60     008     05           $NO:        LD                              ; NO — LOAD COUNT IN ACC.
61     009     5F                      AISC        –1                  ; SUBTRACT ONE.
62     00A     48           $TS2:       RET                             ; MANTISSA IS A 0! RETURN
63     00B     06                      X                               ; STORE CHARACTERISTIC.
64     00C     C3                      JP          $LP1                ; DO IT AGAIN!
65
66
67
68
69
70                          ; 2 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
71                          ; PROGRAM: SDB2, SHLR.
72
```

TL/DD/6942–5

FIGURE 6

The program shown develops the $\log_2$ of any even power of 2 by shifting and testing as previously described. Examine what happens to a value of X that is not an even power of 2. In *Figure 7,* the number 25 is converted to a base 2 log.

$$25_{10} = 00011002_2$$

Shift left until carry = 1

Characteristic	Carry	Mantissa	Log_2
0100	1	10010000	0100.10010000

Figure 7. Straight Line Approximation of $Base_2$ Log

The resulting number when viewed as an integer characteristic and a fractional mantissa is 4.5625_{10}. The fraction 0.5625 is a straight line approximation of the logarithmic curve between the correct values for the $base_2$ logs of 2^4 and 2^5. The accuracy of this approximation is sufficient for many applications. The error can be corrected, as will be seen later in this discussion, but for now let's look at the problem of exponents or the conversion to an antilog.

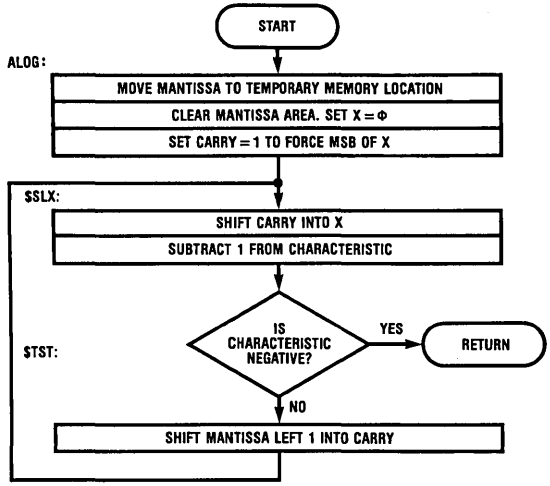
To reconstruct the original value of X, find the antilog, requires only restoration of the most significant bit and then its alignment with the power of 2 position indicated by the characteristic. In the example, approximation ($\log_2$ 25 = 0100.1001) restoration of MSB can be accomplished by shifting the mantissa (only) one position to the right. In the process a one is shifted into the MSB position.

Approximation of Log_2 X	Restoration of MSB
Char. Mantissa	Char. Mantissa
0100.10010000	0100.11001000

The value of the characteristic is 4 so the mantissa must be shifted to the right until MSB is aligned with the 2^4 position.

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

The completion of this operation restores the value of X (X = 25) and is the procedure used to find an antilog. *Figure 8* is a flow chart for finding an antilog using this procedure. Ths implementation in source code is shown in *Figure 9.*



TL/DD/6942-3

FIGURE 8. Flow Chart for Conversion to Antilog

```
COP CROSS ASSEMBLER PAGE 3
LOGS

                                . FORM          ; ------> CONVERT TO ANTILOG <------ ;
                                ; THE FOLLOWING SUBROUTINE CONVERTS THE STRAIGHT LINE
                                ; THE APPROXIMATION OF A BASE 2 LOGARITHM TO ITS CORRESPONDING
                                ; ANTILOG. UPON EXIT FROM THE ROUTINE THE CONTENTS OF CH
                                ; WILL BE EQUAL TO THE HEXADECIMAL VALUE OF 'ΦF .
                                        . LOCAL
00D  A4   ALOG:   JSRP    SDB2          ; SET ACC TO 0.
00E  00   CLRA                          ; CLEAR MANTISSA AREA.
00F  36           X       03            ; AND MOVE MANTISSA TO
010  34           XIS     03            ; TEMPORARY STORAGE.
011  00           CLRA                  ; LEAVE POINTER AT LO
012  36           X       03            ; ORDER OF MANTISSA.
013  37           XDS     03
014  22           SC                    ; RESTORE MSB OF X.
015  D8           JP      $SLX
01   A9   $SLM:   JSRP    SHLR          ; SHIFT REMAINDER
                                        ; LEFT INTO CARRY.
017  A3           JSRP    SDR2          ; MOVE BACK 2 DIGITS.
018  AA   $SLX:   JSRP    SHLC          ; SHIFT X LEFT 1.
019  05           LD                    ; LOAD CHARACTERISTIC.
01A  5F   $TST:   AISC    -1            ; CHARACTERISTIC -1.
01B  48   $LST:   RET                   ; IF NO CARRY — FINIS.
01C  36           X       03            ; STORE REMAINDER AND MOVE
                                        ; DOWN ONE REGISTER.
01D  A4           JSRP    SDB2          ; MOVE BACK 2 DIGITS.
01E  D6           JP      $SLM          ; DO IT AGAIN.

                                ; 4 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
                                ; PROGRAM: SDB2, SDR2, SHLR, SHLC.
```

TL/DD/6942–6

FIGURE 9

Using the linear approximation technique just described, some error will result when converting any value of X that is not an even power of 2.

Figure 10 contains a table of correct base 2 logarithms for values of X from 1 through 32 along with the error incurred for each when using linear approximation. Notice that no error results for values of X that are even powers of 2. Also notice that the error incurred for multiples of even powers of 2 of any given value of X is always the same.

Value of X	Error
5	0.12
$2 \times 5 = 10$	0.12
$4 \times 5 = 20$	0.12
3	0.15
$2 \times 3 = 6$	0.15
$4 \times 3 = 12$	0.15
$8 \times 3 = 24$	0.15

X	Hexadecimal Log Base	Linear Approximation of Log Base 2	Error Hexadecimal	$E_M - 1 + \frac{EM - EM - 1}{2}$
1	0.00	0.00	0.00	
2	1.00	1.00	0.00	
3	1.95	1.80	0.15	
4	2.00	2.00	0.00	
5	2.52	2.40	0.12	
6	2.95	2.80	0.15	
7	2.CE	2.C0	0.0E	
8	3.00	3.00	0.00	
9	3.2B	3.20	0.0B	
10	3.52	3.40	0.12	
11	3.75	3.60	0.15	
12	3.95	3.80	0.15	
13	3.B3	3.A0	0.13	
14	3.CE	3.C0	0.0E	
15	3.E8	3.E0	0.08	
16	4.00	4.00	0.00	
17	4.16	4.10	0.06	0.03
18	4.2B	4.20	0.0B	0.09
19	4.3F	4.30	0.0F	0.0D
20	4.52	4.40	0.12	0.11
21	4.67	4.50	0.17	0.15
22	4.75	4.60	0.15	0.16
23	4.87	4.70	0.17	0.16
24	4.95	4.80	0.15	0.16
25	4.A4	4.90	0.14	0.15
26	4.B3	4.IA0	0.13	0.14
27	4.C1	4.B0	0.11	0.12
28	4.CE;	4.C0	0.0E	0.10
29	4.DB	4.D0	0.0B	0.0D
30	4.E8	4.E0	0.08	0.0A
31	4.F4	4.F0	0.04	0.06
32	5.00	5.00	0.00	0.02
33		5.1-		

FIGURE 10. Error Incurred by Linear Approximation of Base 2 Logs

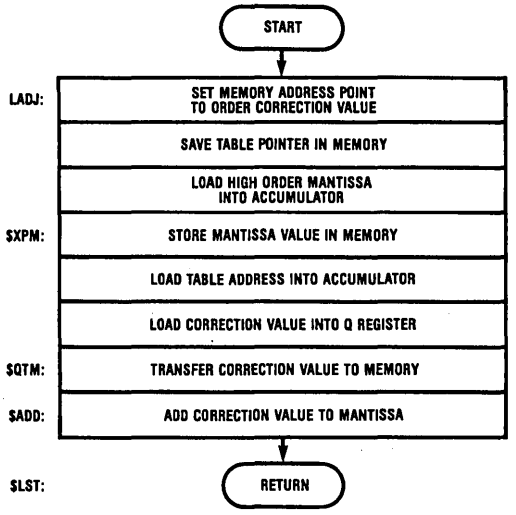
An error that repeats in this way is easily corrected using a look-up table. The greatest absolute error will occur for the least value of X not an even power of 2, X = 3, is about 8%. A 4 point correction table will eliminate this error but will move the greatest uncompensated error to X = 9 where it will be about 4%. This process continues until at 16 correction points the maximum error for the absolute value of the logarithm is less than 1 percent. This can be reduced to 0.3 percent by distributing the error. Interpolated error values are listed in *Figure 10* and are repeated in *Figure 11* as a binary table.

High Order 4 Mantissa Bits	Binary Correction Value	Hexadecimal Correction Value
0000	0000 0000	0 0
0001	0000 1001	0 9
0010	0000 1101	0 3
0011	0001 0001	1 1
0100	0001 0101	1 5
0101	0001 0110	1 6
0110	0001 0110	1 6
0111	0001 0110	1 6
1000	0001 0101	1 5
1001	0001 0100	1 4
1010	0001 0010	1 2
1011	0001 0000	1 0
1100	0000 1101	0 D
1101	0000 1010	0 A
1110	0000 0110	0 6
1111	0000 0010	0 2

FIGURE 11. Correction Table for L_2 X Linear Approximations

Notice in *Figure 10* that left justification of the mantissa causes its high order four bits to form a binary sequence that always corresponds to the proper correction value. This works to advantage when combined with the COP400 LQID instruction. LQID implements a table look-up function using the contents of a memory location as the address pointer. Thus we can perform the required table look-up without disturbing the mantissa.

Figure 12 is the flow chart for correction of a logarithm found by linear approximation. *Figure 13* is its implementation in COP400 assembly language. Notice that there are two entry points into the program. One is for correction of logs (LADJ:), the other is for correction of a value prior to its conversion to an antilog (AADJ:).



TL/DD/6942–4

FIGURE 12. Flow Chart for Correction of a Value Found by Straight Line Approximation

```
COP CROSS ASSEMBLER     PAGE:  4
LOGS

  110                          . FORM      ; ------> ADJUST VALUE OF LOGARITHM <------ ;
  111
  112                                      . LOCAL
  113
  114
  115                          ; THE FOLLOWING TABLE IS USED DURING THE CORRECTION OF VALUES
  116                          ; FOUND BY STRAIGHT LINE APPROXIMATION. IT IS PLACED HERE IN
  117                          ; ORDER TO ALIGN ITS BEGINNING ELEMENT WITH A ZERO ADDRESS AS
  118                          REQUIRED BY THE LQID INSTRUCTION.
  119
  120  01F   44                           NOP                                  ; REGISTER WITH ZERO ADDRESS.
  121  020   03                 TPLS:      . WORD      03,09,0D,011
       021   09
       022   0D
       023   11
  122  024   15                 . WORD     015,016,016,016
       025   16
       026   16
       027   16
  123  028   15                           . WORD      015,014,012,010
       029   14
       02A   12
       02B   10
  124  02C   0D                           . WORD      0D,0A,06,02
       02D   0A
       02E   06
       02F   02
  125
  126                          ; THE FOLLOWING SUBROUTINE ADJUSTS THE VALUE OF A BASE 2
  127                          ; LOGARITHM FOUND BY STRAIGHT LINE APPROXIMATION. THE
  128                          ; CORRECTION TERMS ARE TAKEN FROM THE TABLE ABOVE. THE
  129                          ; SUBROUTINE HAS 2 ENTRY POINTS:
  130                          ;
  131                          ;           LADJ: — ADJUSTS A VALUE DURING CONVERSION TO A LOG
  132                          ;
  133                          ;           AADJ: — ADJUSTS A VALUE DURING CONVERSION TO ANTILOG
  134                          ;
  135                          ; THE CARRY FLAG IS SET UPON ENTRY TO DISTINGUISH BETWEEN LOG
  136                          ; (C = 1) AND ANTILOG (C = 0) CONVERSIONS. DURING A LOGARITHM
  137                          ; CONVERSION THE VALUE FOUND IN THE ABOVE TABLE IS ADDED TO
  138                          ; THE MANTISSA. DURING AN ANTILOG CONVERSION THE VALUE FOUND
  139                          ; IN THE ABOVE TABLE IS SUBTRACTED FROM THE MANTISSA.
  140
  141
  142  030   32      AADJ:      RC                                   ; C = 0 FOR ANTILOG
  143  031   F3                 JP          $LD                      ; CONVERSION.
  144  032   22      LADJ:      SC                                   ; C =  FOR LOG2 ADJ.
  145  033   05      $LD        LD                                   ; MOVE ADDRESS POINTER BACK
  146  034   07                 XDS                                  ; ONE LOCATION.
  147  035   05                 LD                                   ; LOAD CONTENTS OF HI MANTISSA
  148  036   37                 XDS         03                       ; AND STORE IT IN THE LO ORDER
  149  037   06                 X                                    ; OF THE TEMP MEMORY LOCATION.
  150  038   00                 CLRA                                 ; SET TABLE POINTER
  151  039   52                 AISC        TBL                      ; (ACC) TO TABLE ADDRESS.

COP CROSS ASSEMBLER     PAGE:  5
LOGS

  152  03A   BF                LQID                                 ; LOAD CORRECTION VALUE TO Q.
  153  03B   332C    $GTM:      CQMA                                 ; TRANSFER Q REGISTER
  154  03D   04                XIS                                  ; CONTENTS TO MEMORY.
  155  03F   07                XDS
  156  03F   20                SKC                                  ; ANTILOG?

  157  040   80                JSRP        COMP                     ; YES — COMPLIMENT.
  158  041   98      $ADD:      JSRP        ADRO                     ; ADD CORRECTION VALUE
  159                                                               ; TO MANTISSA.
  160  042   35                LD          03                       ; SET POINTER TO
  161  043   48      $LST:      RET         ; CHARACTERISTIC AND
  162                                                               ; RETURN.
  163
  164                          ; 2 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
  165                          ; PROGRAM: COMP, ADRO
  166
  167          0020                       V1 = TPLS&OFF
  168          0002                       TBL = V1/16
  169
  170
  171
```

TL/DD/6942–7

FIGURE 13

Subroutines Used by the Log and Antilog Programs

```
COP CROSS ASSEMBLER    PAGE:  6
LOGS

 172                                    . FORM

 173          0080      . PAGE 02                  ; ------> SUBROUTINES <------ ;
 174
 175                    ; THE FOLLOWING ROUTINES RESIDE ON THE SUBROUTINE PAGE. THEY
 176                    ; ARE CALLED BY THE LOGS PROGRAM BUT ARE GENERAL PURPOSE IN
 177                    ; NATURE AND FUNCTION AS UTILITY ROUTINES.
 178
 179
 180
 181                                    ; ------> COMPLEMENT 8 BITS <------ ;
 182
 183                                    . LOCAL
 184
 185                    ; THIS ROUTINE FORMS IN MEMORY THE 2'S COMPLEMENT OF THE TWO
 186                    ; ADJACENT DIGITS IDENTIFIED BY THE ADDRESS POINTER. THE
 187                    ; CONTENTS OF THE ADDRESS POINTER ARE NOT ALTERED.
 188
 189                    ; THERE ARE TWO ENTRY POINTS:
 190                    ;
 191                    ; COP: COMPLEMENT 8 BITS.
 192                    ;
 193                    ; CMPE: EXTEND THE COMPLEMENT TO AN ADDITIONAL 8 BITS
 194                    ;
 195
 196  080   22          COMP:     SC
 197  081   00          CMPE:     CLRA                          ; SET MINUEND = 0
 198  082   06                    X                             ; AND STORE IN MEMORY.
 199  083   10                    CASC
 200  084   44                    NOP                           ;
 201  085   04                    XIS                           ;
 202  086   00                    CLRA                          ; SET MINUEND = 0
 203  087   06                    X                             ; AND STORE IN MEMORY.
 204  083   10                    CASC                          ;
 205  089   44                    NOP
 206  08A   04                    XIS                           ;
 207  08B   44                    NOP                           ; AVOID SKIP IF DIGIT 15.
 208  08C   A4                    JP        SDB2                ; RETURN THRU SDB2
 209                                                          ; TO RESTORE POINTER.
 210
 211
 212
 213                    ; ------> ADD 8 BITS IN ADJACENT REGISTERS <------ ;
 214
 215                                    . LOCAL
 216
 217
 218
 219                    ; THIS ROUTINE ADDS TWO BINARY DIGITS (8 BITS) FROM ANY REGISTER
 220                    ; TO THE CORRESPONDING TWO BINARY DIGITS IN EITHER REGISTER
 221                    ; IMMEDIATELY ADJACENT. THERE ARE THREE ENTRY POINTS:
 222                    ;
 223                    ;              LADR: — RESET CARRY AND ADD 2 DIGIT PAIRS
```

TL/DD/6942-8

```
COP CROSS ASSEMBLER     PAGE:  7
LOGS

                                ;               LADD: — ADD 2 DIGIT PAIRS WITH UNMODIFIED CARRY
                                ;               ADD1: — ADD 2 SINGLE DIGITS WITH UNMODIFIED CARRY
                                ;
                                ;
                                ;
                                ;
 08D   32       LADR:           RC                                  ; RESET CARRY PRIOR TO ADD.
 08E   15       LADD:           :D              01                  ; LD ADDEND AND MOVE TO ADJ REG
 08F   30                       ASC                                 ; ADD AUGEND.
 090   44                       NOP                                 ; AVOID CARRY!
 091   14                       XIS             01                  ; STORE SUM AND MOVE TO ADDEND
 092   15       ADD1:           LD              01                  ; REPEAT PROCESS
 093   30                       ASC                                 ; FOR
 094   44                       NOP                                 ; HIGH ORDER
 095   14                       XIS             01                  ; DIGIT.
 096   44                       NOP                                 ; AVOID SKIP IF DIGIT 15.
 097   48       $LST:           RET                                 ; FINISHED — RETURN!!!!
                                ;
                                ;
                                ;
                                ;
                                ;               ;------> ADD 8 BITS IN OPPOSITE REGISTERS <------ ;
                                ;
                                ;               . LOCAL
                                ;
                                ;
                                ;
                                ; THIS ROUTINE ADDS TWO BINARY DIGITS (8BITS) FROM ANY REGISTER
                                ; TO THE CORRESPONDING TWO BINARY DIGITS IN EITHER REGISTER
                                ; DIRECTLY OPPOSITE. THERE ARE THREE ENTRY POINTS:
                                ;
                                ;               ADR0: — RESET CARRY AND ADD 2 DIGIT PAIRS
                                ;               ADD0: — ADD 2 DIGIT PAIRS WITH UNMODIFIED CARRY
                                ;               AD01: — ADD 2 SINGLE DIGITS WITH UNMODIFIED CARRY
                                ;
                                ;
                                ;
                                ;
 098   32       ADR0:           RC                                  ; RESET CARRY PRIOR TO ADD.
 099   35       ADD0:           LD              03                  ; LD ADDEND AND MOVE TO OPP REG
 09A   30                       ASC                                 ; ADD AUGEND.
 09B   44                       NOP                                 ; AVOID CARRY!
 09C   34                       XIS             03                  ; STORE SUM AND MOVE TO ADDEND.
 09D   15       AD01:           LD              01                  ; REPEAT PROCESS
 09E   30                       ASC                                 ; FOR
 09F   44                       NOP                                 ; HIGH ORDER
 0A0   34                       XIS             03                  ; DIGIT.
 0A1   44                       NOP                                 ; AVOID SKIP IF DIGIT 15.
 0A2   48       $LST:           RET                                 ; FINISHED — RETURN!!!!
                                ;
                                ;
                                ;
                                ;               ;------> SET DIGIT ADDRESS BACK TWO <------ ;
                                ;
```

TL/DD/6942–9

```
COP CROSS ASSEMBLER     PAGE:  8
LOGS

                                        .LOCAL

                        ;THIS ROUTINE SUBTRACTS 2 FROM THE CONTENTS OF THE
                        ;DIGIT POINTER (B REGISTER). THE CONTENTS OF THE
                        ;ACCUMULATOR ARE LOST IN THE PROCESS. THE USE OF
                        ;SDB2 ALLOWS ADDRESSING WITHIN THE LOGS SUB
                        ;ROUTINE TO BE RELATIVE TO THE CONTENTS OF THE
                        ;ADDRESS POINTER (B REGISTER) UPON ENTRY.
                        ;SDB2 IS COMMONLY USED IN BYTE OPERATIONS TO RESTORE THE
                        ;DIGIT POINTER TO THE LOW ORDER POSITION.
                        ;THERE ARE TWO ENTRY POINTS:
                        ;
                        ;SDR2:       SET DIGIT ADDRESS BACK 2 AND MOVE TO OPPOSITE REGISTER.
                        ;
                        ;SDB2: SET DIGIT ADDRESS BACK 2 RETAINING PRESENT REGISTER.


0A3  35     SDR2:       LD      03          ;MOVE TO OPPOSITE REGISTER.
0A4  4E     SDB2:       CBA                 ;PLACE DIGIT COUNT IN ACC.
0A5  5E                 AISC    -2          ;SUBTRACT 2.
0A6  44                 NOP                 ;SHOULD ALWAYS SKIP.
0A7  50                 CAB                 ;PUT DIGIT COUNT BACK.
0A8  48                 RET                 ;FINISHED — RETURN!!


                        ;——→ SHIFT LEFT ←—— ;

                                        .LOCAL

                        ;THIS ROUTINE SHIFTS LEFT THE CONTENTS OF TWO MEMORY
                        ;LOCATIONS ONE BIT. THERE ARE THREE ENTRY POINTS:

                        ;                   SHLR: RESETS THE CARRY BEFORE SHIFTING
                        ;                         IN ORDER TO FILL THE LOW ORDER
                        ;                         BIT POSITION WITH A 0.
                        ;
                        ;                   SHLC: SHIFTS THE STATE OF THE CARRY INTO
                        ;                         THE LOW ORDER BIT POSITION.
                        ;
                        ;                   SHL1: SHIFTS LEFT THE CONTENTS OF ONLY
                        ;                         ONE MEMORY LOCATION. THE STATE
                        ;                         OF THE CARRY IS SHIFTED INTO THE
                        ;                         LOW ORDER POSITION OF MEMORY.


0A9  32     SHLR:       RC                  ;CLEAR CARRY PRIOR TO SHIFT.
0AA  05     SHLC:       LD                  ;LOAD FIRST MEM DIGIT.
0AB  30                 ASC                 ;DOUBLE IT.
0AC  44                 NOP                 ;AVOID SKIP.
0AD  04                 XIS                 ;STORE SHIFTED DIGIT.
0AE  05     SHL1:       LD                  ;LOAD NEXT MEM DIGIT.
0AF  30                 ASC                 ;DOUBLE IT TOO.

COP CROSS ASSEMBLER     PAGE:  9
LOGS

0B0  44                 NOP                 ;AVOID SKIP, IF ANY
0B1  04                 XIS                 ;STORE SHIFTED DIGIT.
0B2  48     $LST:       RET                 ;FINISHED — RETURN!


                        END
```

TL/DD/6942–10

L-Bus Considerations

National Semiconductor
COP Brief 4

L-BUS CONSIDERATIONS

Users of the COP400 family of microcontrollers should be aware that certain outputs exhibit peculiarities that preclude their use as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. All family members excluding the COP410L and COP411L may generate false states on L_0–L_7 during the execution of the CAMQ instruction. *Figure 1* contains a short program to illustrate this.

```
START:
        CLRA            ;ENABLE THE Q
        LEI   4         ;REGISTER TO L LINES
        LBI   TEST
        STII  3
        AISC  12
LOOP:
        LBI   TEST      ;LOAD Q WITH X'C3
        CAMQ
        JP    LOOP
```

FIGURE 1. Glitch Test Program

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is outpout on L_0, L_1, L_6, L_7, and logic lows on L_2–L_5 via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L_0, L_1, L_6, L_7, and positive glitches on L_2–L_5. Glitch durations are under 2 microseconds, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines. The user should expeience no difficulty interfacing with other COP420 outputs such as G_0–G_3 and D_0–D_3 to edge sensitive components.

Software and Opcode Differences in the COP444L Instruction Set

National Semiconductor
COP Brief 5

The COP444L is essentially a COP420L with double RAM and ROM. Because of this increased memory space certain instructions have expanded capability in the COP444L. Note that there are no new instructions in the COP444L and that all instructions perform the same operations in the COP444L as they did in the COP420L. The expanded capability is merely to allow appropriate handling of the increased memory space. The affected instructions are:

JMP	a	(a = address)
JSR	a	(a = address)
LDD	r,d	(r,d = RAM address Br,Bd)
XAD	r,d	(r,d = RAM address Br,Bd)
LBI	r,d	(r,d = RAM address Br,Bd; only two byte form of the instruction affected)
XABR		

The JMP and JSR instructions are modified in that the address a may be anywhere within the 2048 words of ROM space. The opcodes are as follows:

JMP | 0110 | 0 | $a_{10:9:8}$ |
| $a_{7:0}$ |

JSR | 0110 | 1 | $a_{10:9:8}$ |
| $a_{7:0}$ |

The LDD, XAD, and two byte LBI are modified so that they may address the entire RAM space. The opcodes are as follows:

LDD | 0110 | 0011 |
| 0 | r | d |

XAD | 0010 | 0011 |
| 1 | r | d |

LBI | 0011 | 0011 |
| 1 | r | d |

The XABR instruction change is transparent to the user. The opcode is not changed nor is the function of the instruction. The change is that values of 0 through 7 in A will address registers in the COP444L—i.e. the lower three bits of A become the Br value following the instruction. In the COP420L, the lower two bits of A became the Br value following an XABR instruction.

Note that those instructions which have an exclusive-or argument (LD, X, XIS, XDS) are not affected. The argument is still two bits of the opcode. This means that the exclusive-or aspect of these instructions works within blocks of four registers. It is not possible to toggle Br from a value between 0 and 3 to a value between 4 and 7 by means of these instructions.

There are no other software or opcode differences between the COP444L and the COP420L. Examination of the above changes indicates that the existing opcodes for those instructions have merely been extended. There is no fundamental change.

RAM Keep-Alive

National Semiconductor
COP Brief 6

A COPS™ application is a small scale computer system and the design of a power shut-down is not trivial. During the time that power is available, but out of the designed operating range, the system must be prevented from doing anything to harm protected data. This will typically involve some type of external protection of timing circuit.

There is an option on the COP420, 420L, and 410L parts called "RAM Keep-Alive" that provides a separate power supply to the RAM area of the chip via the CKO pin. The application of power to the RAM while the remainder of the chip has been powered down via V_{CC} will keep the RAM "alive".

However, the integrity of data in the RAM is not only a function of power but is also influenced by transient conditions as power is removed and reapplied. During power-on, the Power On Reset (POR) circuit will keep transients from causing changes in the RAM states. The condition of power loss will have some probability of data change if external control is not used.

At some point below the minimum operating voltage certain gates will no longer respond properly while others may still be functional until a much lower voltage. During this transition time any false signal could cause a false write to one or more cells. Another effect could be to turn on multiple address select lines causing data destruction.

Testing the rate of data change is very difficult because it must be done on a statistical basis with many turn/on-turn/off cycles. Two factors have a major bearing on the numbers derived by testing. One is to call any change in a related data block a failure, even though more than one bit in that block may have changed (this latter case may well be due to the "address select mode"). The second factor is that without massive instrumentation it is impossible to examine the data after each power cycle. Indeed, to do so might have caused errors!

By running the power cycle for a period of time and then looking for changes, one could overlook multiple changes thus reducing the error rate. This has been minimized by more frequent checking which indicates that the errors are spread out randomly over time.

With a power supply that drops from 4.5 to 2V in approximately 100 ms, the drop-out rate is 1 in 5k to 6k power cycles. Reducing the voltage fall time will cause an improvement in the number of cycles per drop-out. This will reach a limit condition of a very high number (1 per 1 million?) when the power falls within one instruction cycle (4–10 μs for the 420, 15–40 μs for the "L" parts). Attaining very rapid fall time may cause problems due to the lack of decoupling/bypass capacitance. By inserting an electronic switch between the regulator and V_{CC} of the COP chip one might be able to meet this type of fall time. By implication some type of sensing is required to cause the switching.

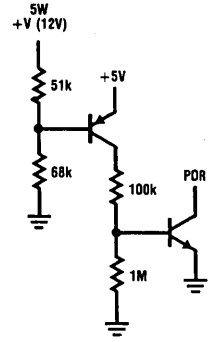
The desirable approach is to force the COP reset input to zero before the voltage falls below 4.5V. This provides a drop out rate of approximately 1 in 50k for the "L" parts and 1 in 100k for the 420. By also stopping the clock of the "L" parts they can achieve a drop-out rate similar to the 420. While not perfect, the number of cycles between data error should be considered with respect to the needs of the application.

The external circuitry to control the chip during the power transition has several implementations each one being a function of the application. The simplest hardware is found in a battery powered (automotive) application. The circuit must sense that the switched 12V is falling (e.g., at some value much below 12V and still greater than 5V). This can be done by using the unswitched 12V as a reference for a divider to a nominal voltage of 8V. As the switched 12V drops below the reference a detector will turn on a clamp transistor to a series switch, the POR, and/or the clock circuit *(Figure 1)*. It should be noted that this draws current during the absence of the switched 12V circuit.

In non-automotive usage a similar circuit can be used where there is a stable reference voltage available to use with the comparator/clamp. Thus a 3.6V rechargable Ni-Cad battery could be used as the reference voltage and V_{RAM} if the appropriate divider is used to level shift to this operating range.

In AC line-powered applications, a similar method could be used with the raw DC being sensed for drop. Another method would be to sense that the line had missed 2–3 cycles either by means of a charge pump or peak detection technique. This will provide the signal to turn on the clamp. One must make this faster than the time to discharge the output capacitance of the power supply, thus assuring that the clamp has performed its function before the supply falls below spec value.

In conclusion, to protect the data stored in RAM during power-off cycle, the POR should go low before the V_{CC} power drops below spec and come up after V_{CC} is within spec. The first item must be handled with an external circuit like *Figure 1* and the latter by an RC per the data sheet.



TL/DD/6946–1

FIGURE 1

Analog to Digital Conversion Techniques With COPS™ Family Microcontrollers

National Semiconductor
COP Note 1
Leonard A. Distaso

TABLE OF CONTENTS

1.0 Introduction

A variety of techniques for performing analog to digital conversion are presented. The COP420 microcontroller is used as the control element in all cases. However, any of the COPS family of microcontrollers could be used with only minor changes in some component values to allow for different instruction cycle times.

Indirect analog to digital converters are composed of three basic building blocks:

- D/A Converter
- Comparator
- Control logic

In a software driven system the D/A converter and comparator are present but the control logic is replaced by instruction sequences. There are a variety of software/hardware techniques for implementing A/D converters. They differ primarily in their approach to the included D/A. There are two primary approaches to the digital to analog conversion which can in turn be divided into a number of sub-categories:

- D/A as a function of weight closures
  - — R/2R ladder
  - — Binary weighted ladder
- D/A as function of time
  - — RC exponential charge
  - — Linear charge/discharge (dual slope)
  - — Pulse width modulation

These techniques should be generally familiar to persons skilled in the electronic art. The objective here is to illustrate the application of these established methods to a low cost system with a COPS microcontroller as the intelligent control element. Circuit configurations are provided as well as the appropriate flow charts and code to implement the function.

Some mathematical and theoretical analysis is presented as an aid to understanding the various techniques and their limits. However, it is not the purpose here to provide a definitive theoretical analysis of the analog to digital conversion process or of the various techniques described.

2.0 Simple Capacitor Charge Time Measurement

2.1 BASIC APPROACH

General

Perhaps the simplest means to perform an analog to digital conversion is to charge a capacitor until the capacitor voltage is equal to the unknown voltage. The capacitor voltage and the unknown are compared by means of a standard analog comparator. The unknown is determined simply by counting, in the microcontroller, the amount of time it takes for the charge on the capacitor to reach a value equal to the unknown voltage. The capacitor voltage is given by the standard capacitor charge equation:

$$V_C = V0 + [V1 - V0][1 - e^{**}(-t/RC)]$$

where: V_C = capacitor voltage
V0 = "dischage voltage" — low level voltage
V1 = high level voltage

The most obvious problem with this method, from the standpoint of software implementation, is the nonlinearity of the

relationship. This can be circumvented in several ways. First of all, a routine to calculate the exponential can be implemented. This, however, usually requires too much code if the exponential routine is not otherwise required in the program. Alternatively, the range of input voltages can be restricted so that only a portion of the capacitor charge curve — which can be approximated with a linear relationship or with some minor straight time curve fitting — is used. Finally, a look up table can be used which will effectively convert the measured time to the appropriate voltage. The look up table has the advantage that all the math can be built into the table, thereby simplifying matters significantly. If arithmetic routines are going to be used, it is clear that the relationship is simplified if V0 is 0V because it then drops out the equation.

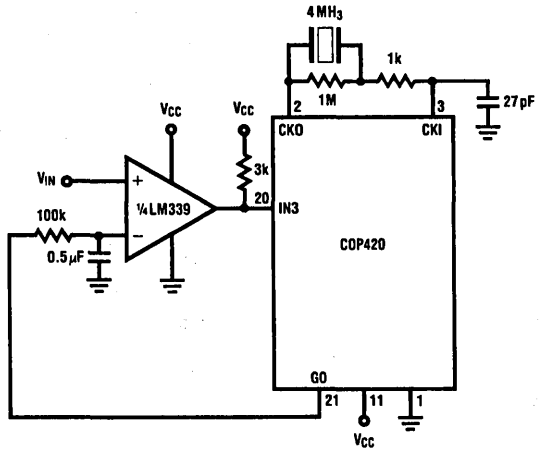
BASIC CIRCUIT IMPLEMENTATION

The circuit in *Figure 1* is the basic implementation of the capacitor charge method of A/D conversion. The selection of input and output used is arbitrary and is dictated by general system considerations. V0 is the "0" level of the G output and V1 is the "1" level of the output. The technique is basically to discharge the capacitor to V0 (which is ideally ground) and then to apply V1 and increment an internal counter until the comparator changes state. The flow chart and code for this implementation are shown in *Figure 2*.

ACCURACY CONSIDERATIONS

The levels reached by the microcontroller output constitute one of the more significant problems with this basic implementation. The levels of V1 and V0 are not V_{CC} and ground as would be desired. The level is defined by the load on the output, the value of V_{CC}, and the device itself. Furthermore, these levels are likely to change from device to device and over temperature. To be sure, the output values will be at least those given in the data sheet, but it must be remembered that those values are minimum high voltages and maximum low voltages. Typically, the high value will be greater than the spec minimum and the low value will be lower than the spec maximum. In fact, with a light load the values will be close to V_{CC} and ground. Therefore, in order to obtain any accurate result for a voltage measurement the exact values of V1 and V0 need to be measured and somehow stored in the microcontroller. Typical values of these voltages can be measured experimentally and an average could be used for final implementation.

The other problem associated with the levels is that the capacitive load on the output line is substantial and far in excess of the values used when specifying the characteristics of the various COP420 outputs. The significant effect of this is that it will take longer than "normal" for the output to reach its maximum value. In addition, it is likely that there will be dips in the output as it rises to its maximum value since the capacitor will start to draw charging current from the output. All of this will be fast relative to the other system times. Still it will affect the result since the level to which the capacitor is attempting to charge is not being applied uniformly and "instantaneously". It can be viewed as though the voltage V1 is bouncing before it stabilizes.



TL/DD/6935–01

Crystal oscillator values chosen to give 4 µs cycle time with divide by 16 option selected on COP 420 CKO/CKI Pins

$V_{CC} = +5V$

FIGURE 1. Basic Capacitor Charge Technique

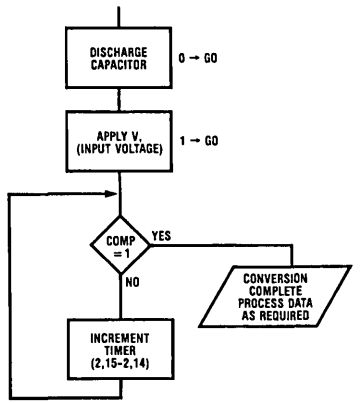
```
        DGI     0       ;TURN OFF Q TO DISCHARGE CAPACITOR
        ; INSERT SOME DELAY TO MAKE SURE CAPACITOR DISCHARGED
        ;USING 12 BIT COUNTER, BUT ONLY UPPER 8 USED IN TABLE
        ;LOOK UP DUE TO ACCURACY OF RC CHARGE METHOD.  THE OTHER
        ;BITS COULD BE USED BUT THE COMPLICATIONS ARE NOT WORTH
        ;THE EFFORT FOR THIS PARTICULAR TECHNIQUE.  ALSO, HERE THE
        ;INPUT RANGE IS RESTRICTED SO THAT THE TOP 3 BITS ARE ZERO

RCAD:   DGI     1       ;TURN ON THE Q LINE
INCR:   LBI     2,13    ;BINARY INCREMENT OF 12 BIT COUNTER
BINPL:  SC              ;LOWER FOUR BITS WILL BE DISCARDED
BINPL1: CLRA            ;ONLY TOP BITS USED IN TABLE LOOK UP
        ASC             ;SPEED WOULD BE IMPROVED IF THE ADD WERE
        NOP             ;STRAIGHT LINE CODED-BUT COSTS MORE CODE
        XIS
        JP      BINPL1
        ININ            ;READ IN3 TO SEE IF COMPARATOR CHANGED
        AISC    8
        JP      END
        CLRA
        JP      INCR
END:    DGI     0       ;TURN OFF THE Q LINE AND DISCHARGE C
        ;DO ARITHMETIC HERE OR LOOK UP TABLE OR WHATEVER IS
        ;REQUIRED--SAMPLE LOOK UP TABLE CONTROL INDICATED BELOW
        ;SAMPLE TABLE WRITTEN CORRECTING FOR THE EXPONENTIAL
        ;RELATIONSHIP.  THE TABLE ALSO INCORPORATES A CONVERSION
        ;TO BCD.  THE VALUE IN THE TABLE IS THE RATIO OF
        ;THE CAPACITOR VOLTAGE V TO THE MAXIMUM VOLTAGE VMAX.
        ;THE NUMBER IS A TWO DIGIT BCD FRACTION.  WE ARE USING
        ;A 5 BIT COUNT IN THIS EXAMPLE.  ADDRESSING ARBITRARILY
        ;SET UP ASSUMING THAT CONTROL CODE IS IN PAGE 0 (OTHER
        ;THAN AT ADDRESS 0) AND THAT THE TABLE THEREFORE IS IN
        ;PAGE 1 (STARTING AT HEX ADDRESS 040).
        ;
        LDI     2,15    ;POINT TO TOP 4 BITS
        XDS             ;TOP 4 IN A,POINTING TO LOWER 4 IN 2,14
        AISC    4       ;THIS MERELY ADJUSTING FOR ADDRESS--NO
                        ;OTHER FUNCTION
        LQID            ;DO THE LOOK UP
        CQMA            ;FETCH THE ADJUSTED VALUE FROM Q
         ; THE ADJUSTED VALUE IS NOW IN A AND M.  FROM THIS POINT MAY
         ;USE THE VALUE IN OTHER CALCULATIONS OR OUTPUT THE INFORMATION,
         ;OR WHATEVER MAY BE REQUIRED BY THE APPLICATION.
        LBI     2,13    ;CLEAR THE COUNTER
        STII    0
        STII    0
        STII    0
        JP      RCAD:   ;JUMP BACK AND REPEAT

        .=X'040         ;SET UP TABLE ADDRESS
        .WORD 000,003,006,008  ;SET UP THE TABLE VALUES
        .WORD 011,014,016,019  ;HERE, COMPENSATED FOR EXPONENTIAL
        .WORD 021,023,026,028  ;AND CONVERTED TO BCD FRACTION
        .WORD 030,032,034,036  ;TABLE VALUE IS RATIO V/VMAX
        .WORD 038,039,041,043
        .WORD 045,046,048,049
        .WORD 051,052,053,055
        .WORD 056,057,059,060
```

TL/DD/6935-55

FIGURE 2A. Typical RC Charge A/D Code



TL/DD/6935-2

FIGURE 2B. Charge Flow Chart

A more general problem is that of the tolerance of RC time constant. The value of the voltage with respect to time is obviously related to the RC value. Therefore, a change in that value will result in a change in the voltage for a given time period t. The graph in *Figure 3* illustrates the effect of a ±10% variation in the RC value upon the voltage measured for a given time t. If one cares to work out the math, it comes out that the error is an exponential relationship in much the same manner as the capacitor voltage itself. The maximum error induced for ±10% RC variation is ±3.9%.

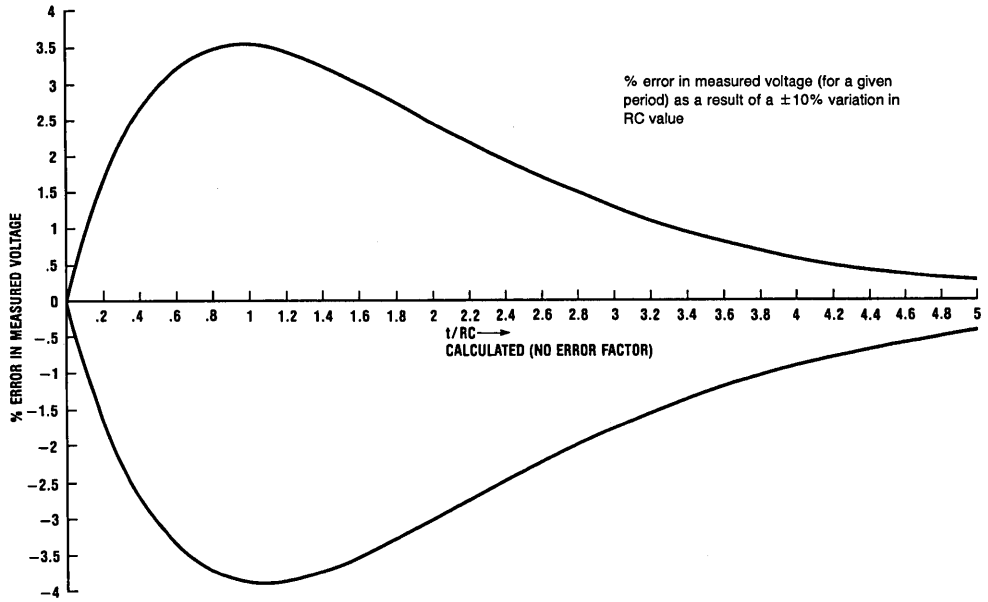
Remember also that we are measuring time. Therefore variation in the RC value will have a direct, linear effect on the time required to measure a given voltage. It is also necessary that the time base for the COP420 be accurate. A variation in the accuracy in the operating frequency of the COP420 will have a direct impact on the accuracy of the result.

Given the errors mentioned so far and assuming that no changes are made in the hardware, the accuracy of the technique then is determined by the resolution of the time measurement. This is improved in two ways: increase the RC time constant so that there is a smaller change in capacitor voltage for a given time period or try to minimize the loop time required to increment the counter. Lengthening the RC time constant is easier but the cost is increased conversion time. The minimum time to increment a 5 to 8 bit binary counter and test an input is 13 cycle times. For a 9 to 12 bit binary counter this minimum time is 17 cycle times. Note also that the minimum time to perform the function does not necessarily correspond to the minimum number of code words required to implement the function. At a cycle time of 4 μs, the 13 cycle times correspond to 52 μs.

2.2 ACCURACY IMPROVEMENTS

Several options are available if it is desired to improve the accuracy of this method. Three such improvements are shown in *Figure 4*. *Figure 4A* is the smallest change. Here a pullup resistor has been added to the G output line and the G line is run open drain internally, i.e., the internal pullup is removed. This improves the "bounce" problem mentioned earlier. The G line will go to the high state and remain there with this setup. However, the addition of the resistor does little more than eliminate the bounce. The degree of improvement is not great, but it is an easy way to eliminate a minor source of error.

Figure 4B is the next step. A 74C04 is used as a buffer. The 74C04 was chosen because of its symmetric output characteristics. Any CMOS gate with such characteristics could be used. The software can easily be adjusted to provide the proper polarity. The COP420 output drives a CMOS gate which in turn drives the RC network. This change does make significant improvements in accuracy. With a light



TL/DD/6935–3

FIGURE 3

load the CMOS gate will typically swing from ground to V_{CC} and its output level is not as likely to be affected by the capacitor discharge.

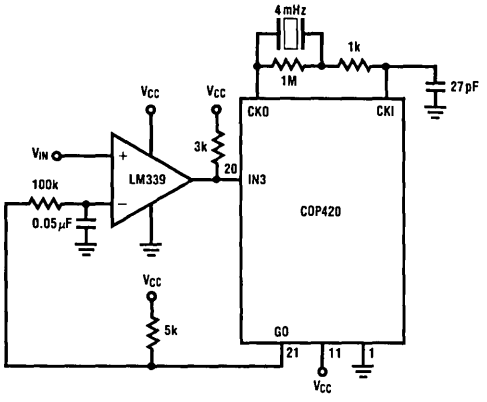
Figure 4C is the best approach, but it involves the greatest component cost. Here two G outputs are controlling analog switches. Ground is connected to the RC network to discharge the capacitor, and a positive reference is used to charge the capacitor. This reference can be any suitable voltage source: zener diodes, V_{CC}, etc. The controlling voltage tolerance is now clearly the tolerance of the reference. Precise voltage references are readily obtainable. *Figure 4C* also shows an analog switch connected directly across the capacitor to speed up the capacitor discharge time. When using this version of the basic scheme, remember to include the 'on' resistance of the analog switch connected to V_{REF} in the RC calculation. Failure to do so will introduce error into the result.

Note that the LM339 is a quad comparator. If these comparators are not otherwise needed in the system, they can be used in much the same manner as the CMOS gate mentioned above. They can be used to buffer the output of the COPS device and to reset the capacitor, or whatever other function is required. This has the advantage of fully utilizing the components in the system and eliminates the need to add another package to the system.

2.3 CONCLUSIONS

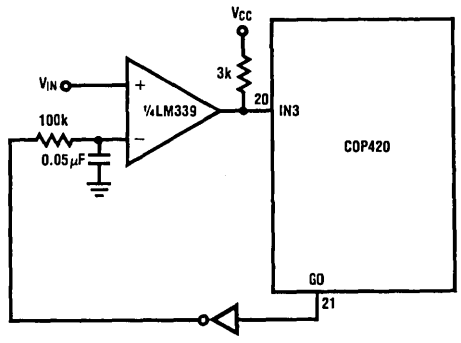
This approach is an inexpensive way to perform an A/D conversion. However, it is not that accurate. With a 10% V_{CC} supply and a 10% tolerance in the RC value and 10% variation in the oscillator frequency the best that can be hoped for is about 25% accuracy. If a 1% reference voltage is used, this accuracy becomes about 15%.

Under laboratory conditions—holding all variables constant and using precise measured values in the calculations—the configuration of *Figure 2* yielded 5 bit accuracy over an input range of 0 to 3.5V. Over the same range and under the same conditions, the circuit of *Figure 4B* yield 7 to 8 bit accuracy. It must be emphasized that these accuracies were obtained under controlled conditions. All variables were held constant and actual measured values were used in all calculations. It is unlikely that the general situation will yield these accuracies unless adjustments are provided and a calibration procedure is used. This could defeat the low cost objective.



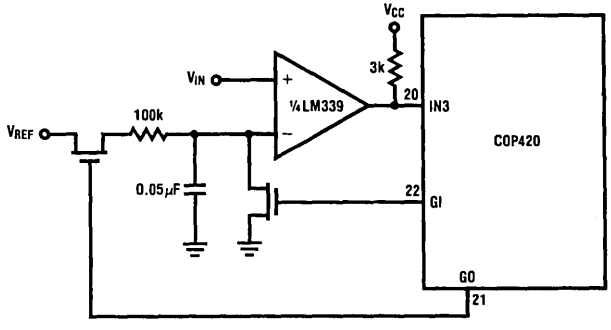
TL/DD/6935-4

A



TL/DD/6935-5

B



TL/DD/6935-6

C

FIGURE 4

3.0 Pulse Width Modulation (Duty Cycle) Technique

3.1 MATHEMATICAL ANALYSIS

The pulse width modulation, or duty cycle, conversion technique is based on the fact that if a repetitive pulse waveform is applied to an RC network, the capacitor will charge to the average voltage of the waveform provided that the RC time constant is sufficiently large relative to the pulse period. See *Figure 5*.

In this technique, the capacitor voltage V_C is compared to the voltage to be measured by means of an analog comparator. The duty cycle is then adjusted to cause V_C to approach the input voltage. The COPS device reads the comparator output and then drives one of its outputs high or low depending on the result, i.e., if V_C is lower than the input voltage, a positive voltage (V1) is applied to charge the capacitor; if V_C is higher than the input voltage, a lower voltage (V0) is applied to discharge the capacitor. Thus the capacitor voltage will seek a point where it varies above and below the input voltage by a small amount. *Figure 6* illustrates the capacitor voltage and the comparator output.

Some mathematical analysis here will be useful to help clarify the technique and to point out its restrictions. Referring to *Figure 6*, we have the following:

$$V_A = V0 + [V_B - V0][e^{**}(-t1/RC)]$$

$$V_B = V_A + [V1 - V_A][1 - e^{**}(-t2/RC)]$$

$$= V1 + [V_A - V1][e^{**}(-t2/RC)]$$

solving for t1 and t2 we have:

$$t1 = -RC\ \ln[(V_A - V0)/(V_B - V0)]$$

$$t2 = -RC\ \ln[(V_B - V1)/(V_A - V1)]$$

let:

$$V_A = V_{IN} - d1$$

$$V_B = V_{IN} - d2$$

substituting the above, the equations for t1 and t2 become:

$$t1 = -RC\ \ln\{[1 - (d1/(V_{IN} - V0))]/[1 + d2/(V_{IN} - V0))]\}$$

$$t2 = -RC\ \ln\{[1 - (d2/(V_{IN} - V1))]/[1 - d1/(V_{IN} - V1))]\}$$

the equations reduce by means of the following assumptions:

1. $d1 = d2 = d$
2. $|V_{IN} - V0| \gg d$
   $|V_{IN} - V1| \gg d$

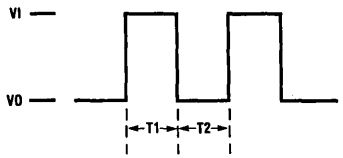
applying these assumptions, we get the following:

$$t1 = -RC\ \ln[(1 + x)/(1 - x)] \text{ where } x = -d/(V_{IN} - V0)$$

$$t2 = -RC\ \ln[(1 + x)/(1 - y) \text{ where } y = d/(V_{IN} - V1)$$

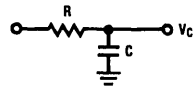
because of the assumptions above, the x and y terms in the preceding equations are less than 1, therefore the following expansion can be used:

$$\ln[(1 + z)/(1 - z)] = 2[z + (z^{**}3)/3 + (z^{**}5)/5 + \ldots]$$



TL/DD/6935-7

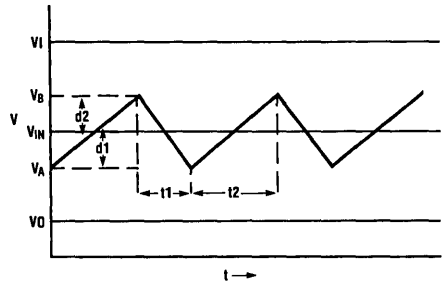
$$V_C = \frac{(V1 - V0) \times T1}{T1 + T2}$$



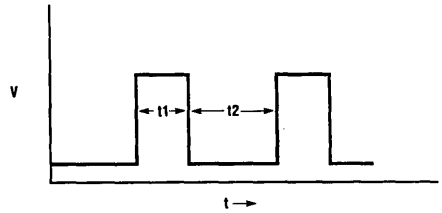
TL/DD/6935-8

FIGURE 5

Capacitor Voltage



Comparator Output



TL/DD/6935-9

TL/DD/6935-10

FIGURE 6

substituting we have:

$t1 = -2RC[x + (x^{**}3)/3 + \ldots]$

$t2 = -2RC[y + (y^{**}3)/3 + \ldots]$

under assumption 2 above, the linear term completely swamps the exponential terms yielding the following result (after substituting back into the equation):

$t1 = 2dRC/V_{IN} - V0)$ $\quad t2 = -2dRC/(V_{IN} - V1)$

therefore:

$$t1/(t1 + t2) = (V1 - V_{IN})/(V1 - V0)$$

$$t2/(t1 + t2) = (V_{IN} - V0)/(V1 - V0)$$

solving for V_{IN}:

$V_{IN} = [t2/(t1 + t2)][V1 - V0] + V0$

or $V_{IN} = V1 - [t1/(t1 + t2)][V1 - V0]$

It follows from the above results that by measuring the times t1 and t2, the input voltage can be accurately determined. As will be seen the restrictions based upon the assumptions above do not cause any serious difficulty.

General Accuracy Considerations

In the preceding calculations it was assumed that the differential output above and below the input voltage was the same. If the comparator output is checked at absolutely regular intervals, and if the intervals are kept as small as possible this assumption can be fairly easily guaranteed—at least to within the comparator offset which is only a few millivolts. As we shall see, this aspect of the technique presents few, if any, difficulties. In addition, there is an RC network at the input of the comparator. The time constant of this network must be long relative to the time between checks of the comparator output. This will insure that the capacitor voltage does not change very much between checks and thereby help to insure that the differences above and below the input voltage are the same.

The next major approximation has to do with the difference between the input voltage and either V1 or V0. We have relied on this difference being much greater than the amount the capacitor voltage changes above and below the input voltage. This approximation allows the nonlinear terms in the logarithmic expansion to be discarded. In practicality, the approximation means that the input voltage must not be "close" to either V1 or V0. Therefore, it becomes necessary to determine how closely the input voltage can approach V1 or V0. It is obvious that the smaller the difference d can be made, the closer the input voltage can approach either reference. The following calculations illustrate the method for determining that difference d. Note, using either V1 or V0 produces the same result. Thus V = V1 = V0.

For at least 1% accuracy

$$x + (x^{**}3)/3 < 1.01x$$

$$\text{therefore } x < 0.173$$

since $x = d/|(V_{IN} - V)|$ we have $d < 0.173|(V_{IN} - V)|$.

Using the same analysis for 0.1% accuracy in the approximation we get $d < 0.0548|(V_{IN} - V)|$. By applying this relationship, the RC time constant can be adjusted so that, within the time interval, the capacitor voltage does not change by more than d V. The user may then select, within reason, how close to the references he can allow the input voltage to go.

The next consideration is really just one of simplification. It is clear that if V0 is zero, it drops out of the first equation and the relationship is simplified. Therefore, it is desirable to use zero volts as the V0 value. The equation then becomes:

$$V_{IN} = V1t2/(t1 + t2).$$

It is obvious by now that the heart of the technique lies in accurately measuring the times t1 and t2. Clearly this requires that the time base of the COP420 be accurate. Short term variations in the COP420 time base will clearly impact the accuracy of the result. In addition to that there is a serious problem in being able to check the comparator output often enough to get any accuracy and resolution out of simply measuring the times t1 and t2. This problem is circumvented by measuring many periods of the waveform. Doing this gives a large average, which improves the accuracy and tends to eliminate any spurious changes. Of course, the trade off is increased time to do the conversion. However if the time is available, the technique becomes restricted only by the accuracy of the external components. Those of the comparator and the reference voltage are most critical.

It is clear from the equation above that the accuracy of the result is directly dependent upon the accuracy of the reference voltage V1. In other words, it is not possible to be more accurate than the reference voltage. If, however, all that is required is a ratio between the input voltage and the reference voltage, the accuracy of the reference will not be a controlling factor provided that the input voltage tracks the reference. This requires that the input voltage be generated from the reference voltage in some form, e.g., a voltage divider with V_{IN} coming off a variable resistance.

Finally, we have noted that the difference d must be small. If the capacitor had to charge or discharge a long way toward V_{IN}, the nonlinearity of the capacitor charge curve would be significant. This therefore requires that the conversion begin with the capacitor voltage close to the input voltage.

Note that the RC value is not part of the equation. Therefore the accuracy of the time constant has no effect on the result as long as the time constant is long relative to the time between checks of the comparator output.

The final point is that the reference voltages, whatever they may be, must be hard sources. Should these voltages vary or drift at all, they will directly affect the result. In those configurations where the references are being switched in and out, the voltage should not change when it is switched into the circuit.

3.2 BASIC IMPLEMENTATION

General

The objective, then, is to measure the times t1 and t2. This is accomplished in the software by means of two counters. One of the two counters counts the t2 time; the other counter counts the total time t1 + t2.

It is necessary to check the comparator output at regular intervals. Thus the software must insure that path lengths

through the test and increment loops are equal in time. Further it is desirable to keep the time required to increment the counters as short as possible. A trade off usually comes into play here. The shortest loop in terms of code required to implement the function is rarely the shortest loop in terms of time required to execute the function. The user has to decide which implementation is best for him. The choice will frequently be governed by factors other than the A/D conversion limits.

It must be remembered that we are now dealing with analog signals. If significant accuracy is required, we are handling very small analog signals. This requires the user to take precautions that are normally required when working with linear circuits, e.g., power supply decoupling and bypassing, lead length restrictions, crosstalk, op amp and comparator stabilization and compensation, desired and undesired feedback, etc. As greater accuracy is sought these factors are more and more significant. It is suggested that the reader refer to the National Semiconductor Linear Applications Handbook and to the data sheets for the various components involved to see what specific precautions should be taken both in general and for a specific device.

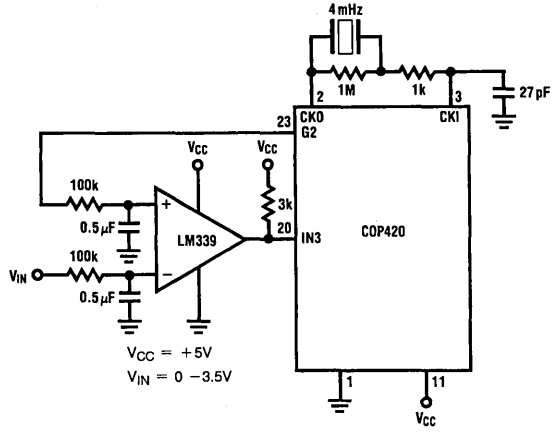
The Base Circuit

Figure 7 shows the diagram for the basic circuit required to implement the duty cycle conversion scheme. The flow chart and code required to implement the function are shown in *Figure 8*. Note that the flow chart and code do not change—except for possible polarity change on output to allow for an inverting buffer—for any of the improvements in accuracy discussed later. The only exception to this is the technique illustrated in *Figure 10* and the variations there are minor.

The code and flow chart in *Figure 8* implement the technique as described above. The large averaging technique is used as it would be too difficult to measure the times t1 and t2 in a single period. The total time, t1 + t2, is the viewing window under complete control of the software. This window is a time equal to the total number of counts, determined by desired accuracy, multiplied by the loop time for a single count. A second counter is counting the t2 time. Special care is taken to insure that all paths through the code take the same length of time since the integrity of the time count is the essence of the technique. The full conversion scheme would use the subroutine in *Figure 8*. Normally the subroutine would be called first just to get the capacitor charged close to the input voltage. The result obtained here would be discarded. Then the routine would be called a second time and the result used as required.

In the configuration in *Figure 7*, there is an RC network in both input legs of the comparator. This is to balance the inputs of the device. For this reason, R1 = R2. C1 is the capacitor whose voltage is being varied by the pulse waveform. C2 is in the circuit only for stabilization and symmetry and is not significant in the result. The comparator tends to oscillate when the + and − inputs are nearly equal without capacitor C2 in the circuit.

As would be expected, the basic circuit has some difficulties. By far the most serious of these difficulties is the output level of the G line. To be sure of the high and low level of this output the levels should be measured. The "1" level will be between the spec minimum of 2.4V and V_{CC} (here assumed to be 5V). The "0" level will be between the 0.4V spec maximum and ground. With light loads, these levels are likely to vary from device to device. Furthermore, we have the same "1" level problem that was mentioned in the simplest technique: the capacitive load is large and the capacitor is charging while the output is trying to go to the high level.



TL/DD/6935–11

FIGURE 7. Basic Duty Cycle A/D

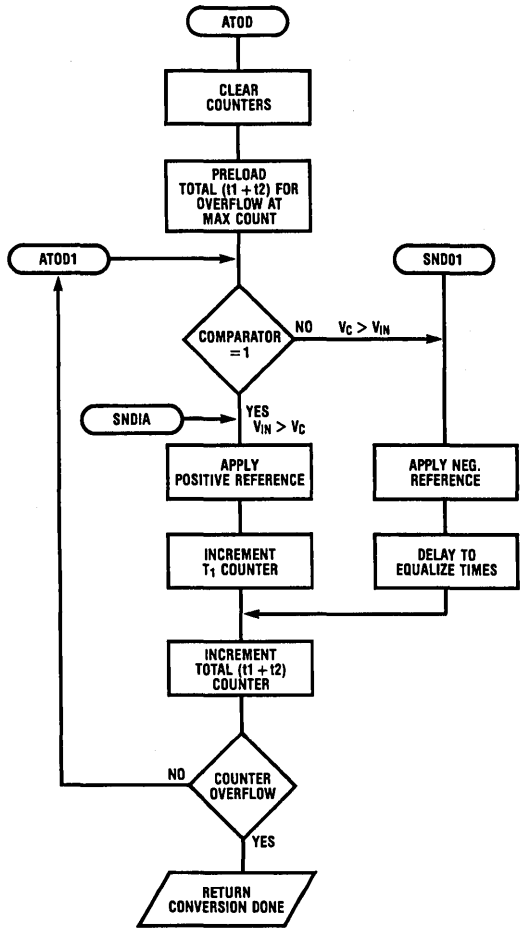
There is also a problem with the low level. When the output goes low, the capacitor begins to discharge through the output device of the COP420. This discharge current has the effect of raising the "0" level and thereby introducing error. Note that we are not talking about large changes in the voltages, especially the low level. Typically, the change will only be a few millivolts but that can translate into a loss of accuracy of several bits.

Under laboratory conditions—holding all variables constant and using precise measured values in the calculations—the circuit of *Figure 7* yielded 5 bit ± 1 bit accuracy over the range of V0 (here measured to be 0.028V) to 3.5V (the maximum specified input voltage for the comparator with V_S = 5V). Increasing the number of total counts had very little effect on the result. In the general case, the basic scheme should not be relied upon for more than 4 bits of accuracy, especially if one assumes that V1 = V_{CC} and V0 = 0. As shall be seen, it is not difficult to improve this accuracy considerably.

```
;ATOD IS THE FULL CONVERSION SCHEME WRITTEN AS A SUBROUTINE
ATOD:   LBI     1,10    ;MAKE SURE COUNTERS CLEARED
        JSRP    CLEAR
        LBI     2,10
        JSRP    CLEAR
        LBI     1,13    ;PRELOAD FOR TOTAL COUNT = 2048
        STII    0
        STII    0
        STII    8
ATOD1:  ININ            ;READ COMPARATOR--INPUT TO 420 = IN3
        AISC    8
        JP      SND01
SND1A:  LBI     3,0     ;USING OMG BELOW TO SAVE STATE OF OTHER G
                ;VALUES IF IT WAS NECESSARY TO DO SO,ELSE USE OGI
        SMB     2       ;VIN > Vc,DRIVE Vc HIGHER
        OMG             ;THIS CODE STRAIGHT LINED FOR SPEED
        SC              ;APPLY POSITIVE REFERENCE
        CLRA            ;INCREMENT THE SUB COUNTER
        LBI     2,13
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP             ;BINARY INCREMENT
        XIS             ;WOULD ELIMINATE THESE 4 WORDS IF 8 BIT
        CLRA            ;COUNTER OR LESS-HERE SET UP FOR UP TO 12 BIT
        ASC             ;COUNTER
        NOP
        X
        JP      TOTAL
SND01:  LBI     3,0
        RMB     2
        OMG
        CLRA
        AISC    10      ;THIS PART OF THE CODE MERELY INSURES THAT
        NOP             ;ALL PATHS THROUGH THE ROUTINE ARE EQUAL IN TI
DLY:    AISC    1
        JP      DLY
TOTAL:  CLRA
        LBI     1,13
        SC
        ASC             ;INCREMENT THE TOTAL LOOP COUNTER
        NOP             ;WHEN OVERFLOW,DONE SO EXIT
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        JP      ATOD2
        RET
ATOD2:  X
        JP      ATOD1
        .PAGE   2
CLEAR:  CLRA
        XIS
        JP      CLEAR
        RET
```

TL/DD/6935-45

FIGURE 8A. Duty Cycle A/D Code



TL/DD/6935–12

FIGURE 8B. Duty Cycle A/D Flow Chart

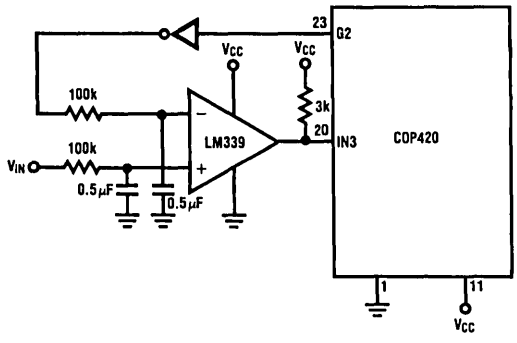
3.3 ACCURACY IMPROVEMENTS

General Improvements

Figure 9 illustrates circuit changes that will make significant improvements in the accuracy of the technique. In *Figure 9A* a CMOS buffer is used to drive the RC network. The output of the COP420 drives the CMOS gate, which here is a 74C04 because of its output characteristics. The main thing that this technique does is to reduce the difficulties with the output levels. Typically, V0 is 0V and V1 is V_{CC}. We also have a "harder" source for the voltages — the levels don't change while the capacitor is charging or discharging. Now, even more clearly than before, the accuracy of V_{CC} is the controlling voltage tolerance. The accuracy of the result will be no better than the accuracy of V_{CC} (for a system requiring absolute accuracy).

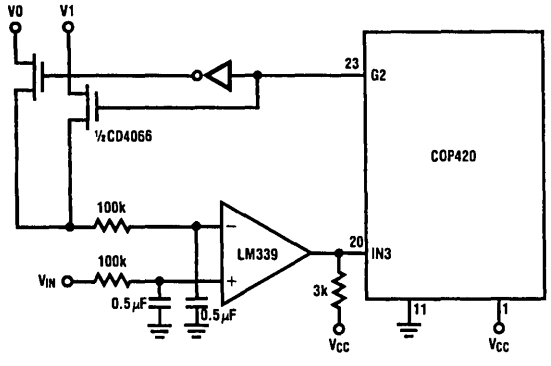
Under laboratory conditions, the circuit of *Figure 9A* yielded the accuracies as indicated below for various total counts. The accuracy increased with the total count until the count exceeded 2048. There was no significant increase in accuracy with this circuit for counts in excess of 2048. (Remember that these results were obtained under controlled conditions). We may then view the results obtained with 2048 counts as the upper limit of accuracy with the circuits of *Figure 9A*. The results were as follows:

Total Count	Resultant Accuracy
512	8 ± 1/2 bits
1024	9 ± 1bits
2048	9 ± 1/2 bits
4096	9 ± 1/2 bits



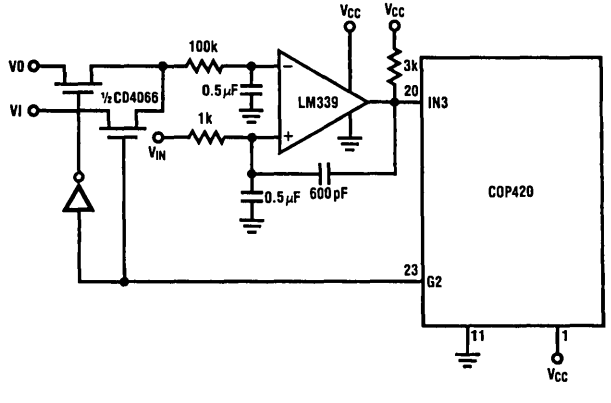
TL/DD/6935–13

A



TL/DD/6935–14

B



TL/DD/6935–15

C

FIGURE 9. Improvements to Duty Cycle A/D

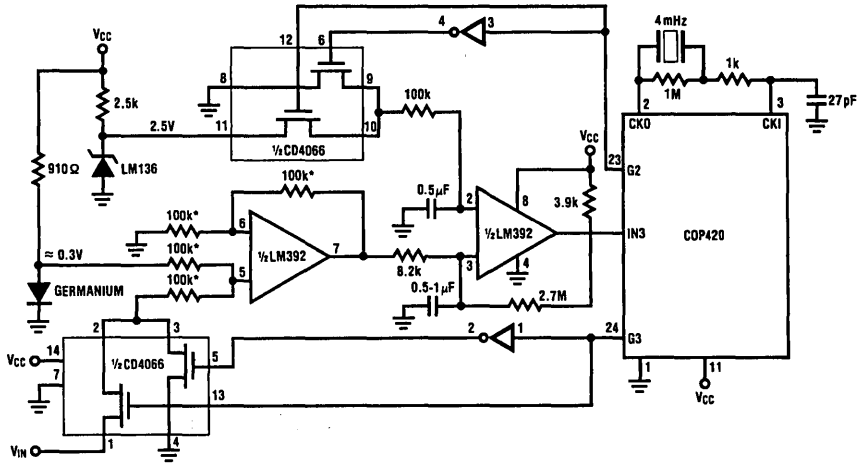
The circuit of *Figure 9B* makes a significant change to improve accuracy. Now the COP420 is controlling analog switches and switching in positive and negative references. Therefore the accuracy of the reference voltages is the controlling factor. Generally this will improve the accuracy over that obtained with *Figure 9A*. With the circuit of *Figure 9B*, with V0 = 1V (negative reference), and V1 = 3V (positive reference), 9 bit accuracy was achieved with a total count of 1024. V0 and V1 were arbitrarily chosen to place the input voltage approximately in the center of the allowable comparator input range with V_S = 5V. Remember, the accuracy of the references is controlling. The result can be no more accurate than the references. Furthermore, these references must be hard sources; i.e., they must not change when they are switched into the circuit as that contributes error into the result.

In *Figure 9C*, capacitive feedback was added to the comparator circuit and the series resistance to V_{IN} was decreased. The feedback added hysteresis and forced the comparator to slew at its maximum rate (significant errors are introduced if the comparator does not change state in a time shorter than the cycle time of the controller). Both of these changes resulted in increased accuracy of the result. With V0 = 0, V1 = 5V (V_{CC}) and V_{CC} held steady at 5.000V, an accuracy of 10 bits ± 1 bit was achieved over the input range of 0 to 3.5V.

It is obviously possible to use any combination of the configurations in *Figure 9* for a given application. What is used will depend on the user and his specific requirements.

Figure 10 illustrates a further refinement of the basic approach. This configuration can be used if greater accuracies are needed. The major change is the addition of a summing amplifier to the circuit for the purpose of adding a fixed offset voltage to the input voltage. This has the effect of moving the input voltage away from the negative reference (which is 0V here). This offset voltage should be stable as the changes in it will directly affect the result. The offset voltage should be chosen so as to place the effective input voltage (the voltage at the comparator input) approximately in the center of the range between the two references. The precise value of the offset is not critical nor is its source. The forward voltage drop across a germanium diode is used as the offset in *Figure 10*, but this offset can be generated in any convenient manner. The forward voltage drop of the germanium diode is approximately 0.3V. Given this and the negative reference of 0V and a positive reference of 2.5V, the input voltage is restricted to a range of 0 to 2V. Therefore, the effective input voltage (at the comparator input) is approximately 0.3V to 2.3V — well within the limits of the two references. The circuit also includes provision for an autozero self calibration procedure.

Note that the resistors in the summing amplifier should be matched. The absolute accuracy of these resistors is not significant, but their accuracy relative to one another can have a significant bearing on the result. The restriction is imposed so that the output of the summing amplifier is exactly the sum of the input voltage and the offset voltage. This requires unity gain through the amplifier and that the



*Resistors should be matched

V_{CC} = +5V

$0 \le V_{IN} \le 2V$

FIGURE 10. Improved Duty Cycle A/D with Autozero

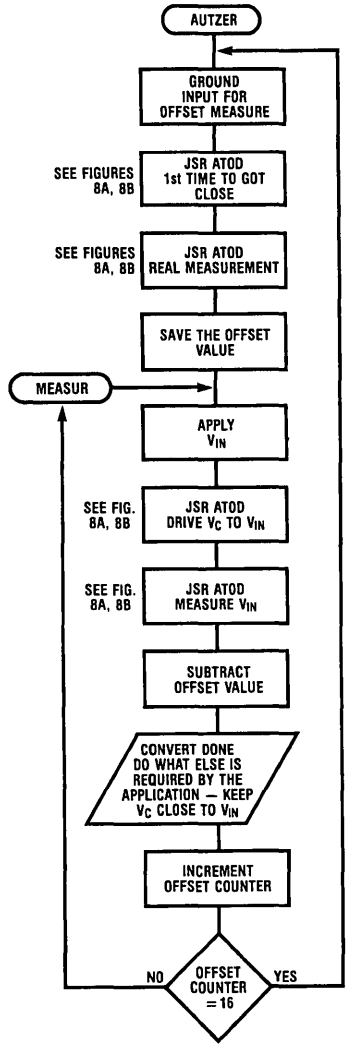
impedance in each summing leg be the same. These effects can become very serious if one is trying for significant accuracy—e.g., if 12 bit accuracy is being sought 1% matching of those resistors can introduce an error of 1% maximum. While 1% accurate is fairly good, it is significantly less than 12 bit accuracy. Related to this effect is a possible problem with the source impedance of the input voltage. If that impedance is significant in terms of its ratio to the summing resistor, errors are introduced just as if the resistors are mismatched. "Significant" is determined in terms of the desired system accuracy and the relative impedance values. The comparator section is using some feedback to provide hysteresis for stability and a low series resistance is used for the input to the comparator.

Most significantly, this configuration allows a true zeroing of the system. Through the additional analog switches shown, the COP420 can easily perform an autozero function by tying the input to ground and measuring the result. Thus the system offsets can be calculated, stored and subtracted from the result. This improves the accuracy and is also more forgiving on the choice of the comparator and op amp selected. Furthermore, the offset can be periodically recomputed by the COP420 thereby compensating for drift in system offsets. Nonetheless, the accuracy of the reference is the controlling factor. It is NOT possible to obtain an absolute (as opposed to ratiometric) accuracy of 12 bits without a reference that is accurate to 12 bits. The LM136 used in *Figure 10* is a 1% reference. Although not inherently accurate to 12 bits, the voltage of the LM136 may be trimmed to exact value by means of a variable resistor. The data sheet of the LM136 illustrates this connection. Under laboratory conditions, the circuit of *Figure 1* yielded 11 bit ±1 bit accuracy with a total count of 4096 over the input range of 0 to 2V. *Figure 11* indicates the flow chart and the code required to implement the technique of *Figure 10*.

```
        ;CODE FOR IMPROVED A TO D PULSE WIDTH METHOD
        ;SEE FIGURE 8A FOR CODE FOR ROUTINE ATOD
        ;
AUTZER: LBI     3,0     ;DO AUTO ZERO, 3,0 CONTAINS G STATUS
        RMB     3       ;SET UP TO GRND INPUT & MEASURE OFFSET
        JSR     ATOD    ;FIRST TIME IS TO GET CLOSE
        JSR     ATOD    ;MEASURE THE OFFSET
        LBI     2,13    ;NOW SAVE THE OFFSET VOLTAGE
XFER:   LD      1               ;SAVE THE OFFSET VALUE IN M3
        XIS     1
        JP      XFER
        LBI     0,0
        JP      INPUT
MEASUR:         ;NOW DO REAL MEASUR(1ST TIME IS OFFSET)
        JSR     ATOD    ;FIRST TIME TO GET CLOSE
        JSR     ATOD    ;NOW REAL MEASUREMENT
        JSRP    BINSUB  ;SUBTRACT THE OFFSET
        ;HAVE THE VALUE AT THIS POINT(IN BINARY)-NOW DO WHAT
        ;THE APPLICATION REQUIRES.  VALUE MUST BE MULTIPLIED
        ;BY (VREF+/TOTAL COUNT) TO GET FINAL VALUE IF SUCH IS
        ;DESIRED
        LBI     1,0     ;INCREMENT COUNTER FOR NEW OFFSET MEASURE
        LD
        AISC    1
        JP      SAVE
        X               ;IS 16TH TIME, MEASURE OFFSET AGAIN
        JP      AUTZER
SAVE:   X
        LBI     3,0
        SMB     3       ;SET BIT SO CAN MEASURE VIN
        JP      MEASUR
        .PAGE   2
BINSUB: LBI     3,13
        SC
BNSUB2: LD      1
        CASC
        NOP
        XIS     1
        JP      BNSUB2
        RET
```

TL/DD/6935-46

FIGURE 11A. Duty Cycle A to D, Improved Method



TL/DD/6935–17

FIGURE 11B. Flow Chart for Improved Duty Cycle A/D

4.0 Dual Slope Integration Techniques

4.1 Mathematical Background

(Some of this background information is taken from National Semiconductor Linear Applications Note AN-155. The reader is referred to that document for other related general information.)

The basic approach of dual slope integration conversion techniques is to integrate a voltage across a capacitor for a fixed time, and then to integrate in the other direction with a known voltage until the starting point is reached. The ratio of the two times then represents the unknown voltage. Some of the math below in conjunction with *Figure 12* will illustrate the approach.

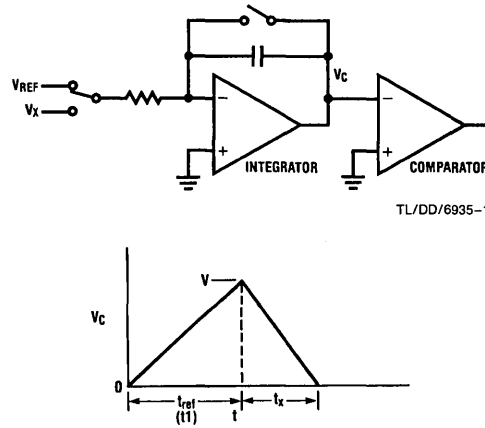


FIGURE 12. Dual Slope Integration—Basic Concept

$$I_X = C\frac{dV}{dt} = V_X/R$$

$$V_X = RC\frac{dv}{dt}$$

$$\int_{\circ}^{T1} V_X dt = \int_{\circ}^{V} RCdV$$

$$V_X T1 = RCV$$

$$V = V_X T1/RC = I_X T1/C$$

Similarly:

$$I_{REF} = C\frac{dV}{dt} = V_{REF}/R$$

$$V_{REF} = RC\frac{dV}{dt}$$

$$\int_{T1}^{T1+T_X} V_{REF} dt = \int_{V}^{\circ} RCdV$$

$$V_{REF}T_X = -RCV$$

$$V = -V_{REF}T_X/RC$$

$$-V_{REF}T_X/RC = V_X T1/RC$$

$$V_X = -V_{REF}T_X/T1$$

Two important facts arise from the preceding mathematics. First of all, there is a linear relationship involved in determining the unknown voltage. Secondly, the negative sign in the final equation indicates that the reference and the unknown, relative to some point (which may be 0V or some bias voltage), have opposite polarity. Thus, if it is desired to measure 0 to +5V, the reference voltage must be −5V. If the input is restricted to 2.5 to 5V, the reference can be 0V as the integrator and comparator are biased at +2.5V (then the 0V is in fact −2.5V relative to the biasing voltage, and the input range is 0 to 2.5V relative to the same bias voltage).

There are some difficulties with dual polarity conversion using the dual slope method. It is clear from the math above that if the input voltage will be dual polarity, it is necessary to have two references—one of each polarity. The midrange biasing arrangement briefly described above eliminates

the need for two different polarities but does not help very much since two references are still required—one at the positive value and one at the bias value. Ground is the other reference. Further, the need to select one of two references further complicates the circuitry involved to implement the approach. Also, the dual requirement brings up a difficulty with the bias currents of the integrator and comparator. They could add to the slope in one polarity and subtract in the other.

The only real operational difficulty in dual slope systems is establishing the initial conditions on the integrating capacitor. If this capacitor is not at the proper initial conditions, accuracy will be severely impaired. *Figure 12* indicates a switch across the capacitor as a means of initializing it. In a software driven system, the initialization can be accomplished by doing two successive conversions. The result of the first conversion is discarded. It is performed only to initialize the capacitor. The second conversion produces the valid result. One need only insure that there is not significant time lapse between the two conversions. They should take place immediately after one another.

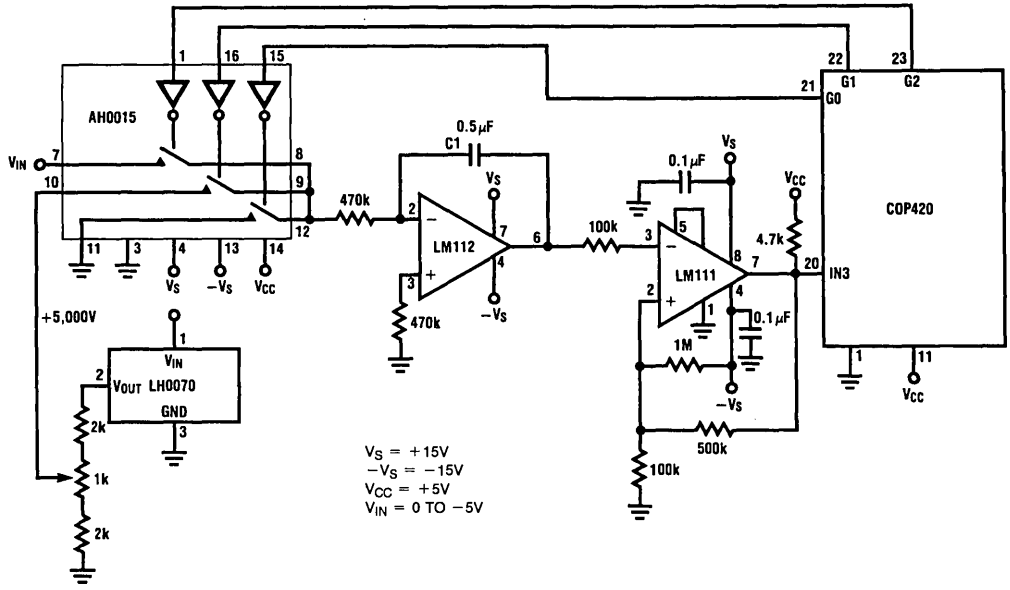
This approach obviously lengthens conversion time but it eliminates many problems. The alternative to this approach of two successive conversions is to take a great deal of care in insuring the initial state of the integrating capacitor and in selecting op amps and comparators with low offsets.

4.2 THE BASIC DUAL SLOPE TECHNIQUE

Figure 13 indicates an implementation of the basic dual slope technique. This is a single polarity system and thus requires only the single reference voltage. The circuit of *Figure 13* is perhaps not the cheapest way to implement such a scheme but it is representative and illustrates the factors that must be considered.

Consider first the means of initializing the integrating capacitor C1. The routine here connects the input to ground and does a conversion on zero volts as a means of initialization. Subsequently—and this is typical of the more usual technique—two conversions are performed. The first conversion is to initialize the capacitor. The second conversion yields the result. Some form of initialization or calibration procedure is required to achieve optimum accuracy from dual slope conversion schemes.

The comparator in this circuit is used in the inverting mode and has positive feedback as recommended in the LM111 data sheet. The voltage reference is the LH0070, which is a 0.01% reference. A resistive voltage divider on the IH0070 creates the 5V value. The use of the voltage divider brings up two difficulties (which can be overcome if the LH0070 is used at its full value, thus eliminating the divider, and the result properly scaled in the microcontroller or series integrating resistor increased). First, the impedance of the reference must be small relative to the series resistance used in the integrator. If this were not the case, the slopes would



TL/DD/6935–20

FIGURE 13. Basic Dual Slope Integration A/D Scheme

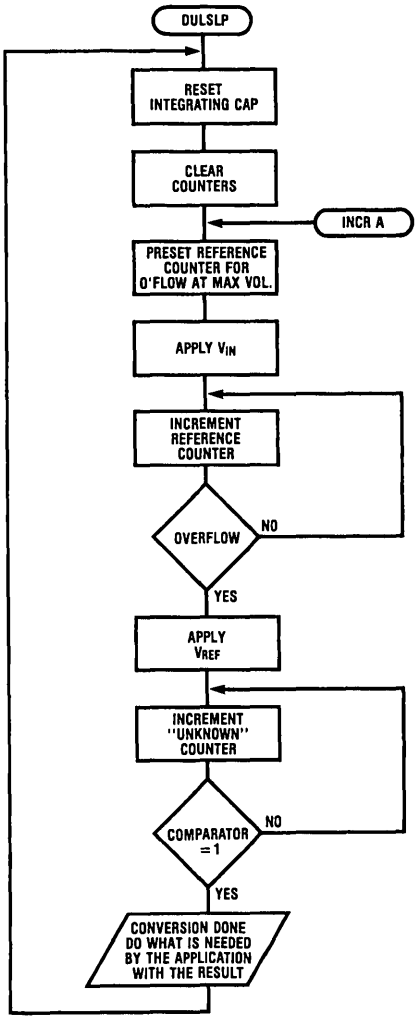
show an effect due to the difference in the R value between the applied reference voltage and the unknown input. (By the same token, the output impedance of the source supplying the unknown must also be small relative to that series integrating resistor). Secondly, the bias currents of the integrator may be such as to affect the reference voltage when it is coming from a simple resistor divider. Both problems are reduced if small resistor values are used in the divider. Note also that current mode switching would reduce the problem as well. It should be pointed out that the errors introduced by these problems are not gross deviations from the expected value. They are small errors that will not make much difference in the majority of applications. They are, however the kind of errors that can make the difference between a system accurate to 10 bits and one accurate to 12 bits (assuming all other factors are the same).

Figure 14 shows the flow chart and code required to implement the basic dual slope technique as shown in *Figure 13*. Under laboratory conditions an accuracy of 12 bits ±1 bit was achieved. The method is slow, with the maximum conversion time equal to 2 x T_{REF}. Notice that the accuracy of V_{CC} and that of the integrating resistor and capacitor are not involved in the accuracy of the result. The accuracy of V_{REF} is, of course, controlling if absolute accuracy—rather than ratiometric accuracy—is desired. The absolute accuracy of the circuit can be no better than the accuracy of the reference. If ratiometric accuracy is all that is required, there is no particular problem. The accuracy is merely relative to the reference. The R and C values do not impact the accuracy because the integration in both directions is being done through the same R and C. Results would be quite different if a different value of R or C was used for one of the slopes.

```
DUALSLP: OGI     1          ;HOLD THE INPUT TO GROUND TO RESET THE
         LBI     2,11       ;INTEGRATING CAPACITOR
         JSRP    CLEAR      ;CLEAR THE COUNTER
         JSR     INCRA      ;TO GET US CLOSE, NEXT READING IS REAL
CLEAR2:  LBI     2,11       ;NOW CLEAR THE COUNTER
         JSRP    CLEAR      ;MAKE SURE COUNTER CLEARED TO ZERO
  ;J,15 = 0 AND START AT 1,13 FOR COUNT = 4096
  ;J,15 = 14 AND START AT 1,12 FOR COUNT = 8192
  ;J,15 = 12 AND START AT 1,12 FOR COUNT = 16384
  ;FOLLOW SAME PATTERN FOR OTHER COUNTS
  ;
MEASUR:  JSR     INCRA      ;RUN THRU THE INCREMENTS
         ; NOW HAVE THE BINARY VALUE, USE IT AS IS OR
         ; MULTIPLY BY (Vref/TOTAL COUNT) TO CREATE THE VOLTAGE
         ; RESULT--THEN CONTINUE WITH THE OPERATION
         LBI     2,11
         JSRP    CLEAR      ;CLEAR THE COUNTER
         JSR     INCRA      ;TO GET CAP CLOSE TO 0 AGAIN
         JP      CLEAR2
 ;FOLLOWING SUBROUTINE INCRA IS THE REAL PART OF THE ROUTINE
 ;CONCERNED WITH THE COUNTING FOR THE CONVERSION.
INCRA:   LBI     1,15       ;R1 IS CLEARED PRIOR TO START
         STII    15         ;PRESET THE COUNTER FOR 4096
         OGI     4          ;APPLY VIN
INCR:    LBI     1,12
         SC
BINAD1:  CLRA
         ASC
         NOP
         XIS
         JP      BINAD1
         NOP                ;2 NOPS TO EQUALIZE TIMES
         NOP
         SKC
         JP      INCR
         OGI     2          ;DONE, NOW APPLY VREF
INCR2:   LBI     2,12       ;COUNT UNTIL COMPARATOR CHANGES
         SC
BINAD2:  CLRA
         ASC
         NOP
         XIS
         JP      BINAD2     ;STRAIGHT LINE THE ADD FOR SPEED
         ININ               ;SAVE WORDS BY USING G
         AISC    8          ;SEE IF IN3=1
         JP      INCR2      ;IN3 IS 0, KEEP COUNTING
OUTPUT:  OGI     1          ;KEEP INPUT AT 0
         RET
```

TL/DD/6935-47

FIGURE 14A. Dual Slope A/D Code



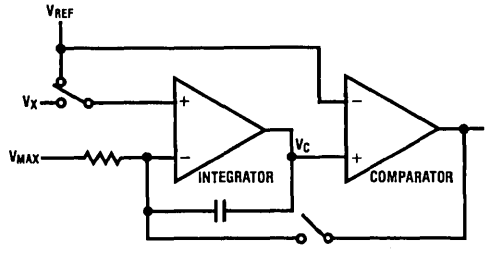
TL/DD/6935-21

FIGURE 14B. Basic Dual Slope A/D Flow Chart

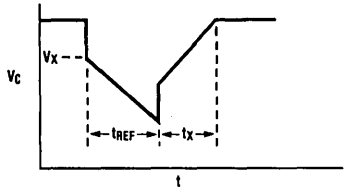
4.3 MODIFIED DUAL SLOPE TECHNIQUE

General

The basic idea of the modified dual slope technique is the same as that of the basic approach. The modified approach eliminates the need for dual polarity references and is also more forgiving in the selection of the op amp and comparator required. *Figure 15* illustrates the basic idea.



TL/DD/6935-22



TL/DD/6935-23

FIGURE 15. Modified Dual Slope — Basic Concept

The math analysis is much the same:

$$I_X = C\frac{dV}{dt} = (V_X - V_{MAX})/R$$

$$V_X - V_{MAX} = RC\frac{dV}{dt}$$

$$(V_X - V_{MAX})T1 = RC$$

$$V = (V_X - V_{MAX})T1/RC$$

Similarly:

$$I_{REF} = C\frac{dV}{dt} = (V_{REF} - V_{MAX})/R$$

$$(V_{REF} - V_{MAX})T_X = -VRC$$

$$V = -(V_{REF} - V_{MAX})T_X/RC$$

$$(V_{MAX} - V_{REF})T_X = (V_X - V_{MAX})T1$$

$$V_X = V_{MAX} + (V_{MAX} - V_{REF})T_X/T1$$

The main difference between this and the basic approach is the offset voltage V_{MAX}. The main restriction is that all input voltage values (V_X) are less than V_{MAX}. It is also apparent that the total count is proportional to the difference between V_{MAX} and V_X. The only significant effect of this is, however, to slightly complicate the arithmetic required to arrive at a value for V_X.

Given that the input voltage V_X is always less than V_{MAX}, the modified dual slope technique is automatic polarity. This fact comes straight out of the equation above. Thus dual polarity references are not required. However, two precise voltages are required: V_{MAX} and V_{REF}. However, the V_{MAX} value can be used for a zero adjust as indicated in *Figure 16*. This means that the V_{MAX} value need not be so precise as it will be adjusted in a calibration procedure to produce a zero output. This adjustment amounts to a compensation for the bias currents and offsets. Thus the COP420 can use the supposed value of V_{MAX} with V_{MAX} later being "tweaked" to give the proper result at zero input. In addition, the initialization loop for the integrating capacitor includes the comparator. Thus the intial condition on the capacitor becomes

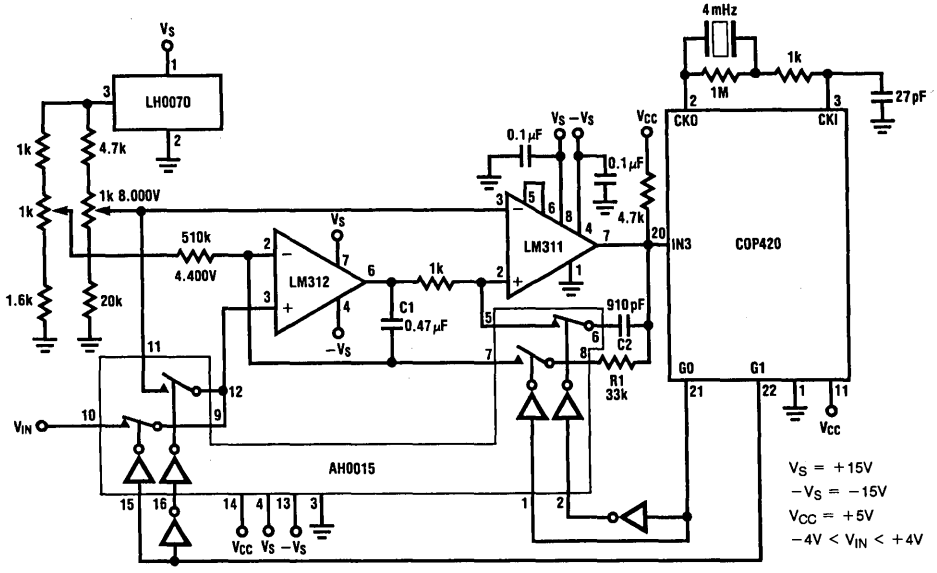
not zero but the sum of the offset voltages of the comparator and op amp. Thus the choice of these components is not critical in a modified dual slope approach.

An Example of the Modified Dual Slope Approach

Figure 16 illustrates an implementation of the modified dual slope technique. The system is calibrated by holding V_{IN} to ground and then adjusting V_{MAX} for a "0" result. Capacitor C1 is the integrating capacitor. Capacitor C2 is used only to cause a rapid transition on the comparator output. C2 is especially useful if an op amp is being used as the comparator stage. Resistor R1 is just part of the capacitor initializing loop. An LH0070 is being used to generate the reference voltage and the V_{MAX} value. The discussion previously about these being hard sources is equally relevant here. In fact, this problem was much more significant in this particular implementation and made the difference between a 10 and 12 bit system. As shown, the technique was accurate to 10 bits. Another bit was obtained when the V_{MAX} and V_{REF} values were buffered. It must be remembered that when trying to achieve accuracies of this magnitude board layout, parts placement, lead length, etc. become significant factors that must be specifically addressed by the user.

There are some other considerations in using this technique. The amount of time required to count the specified number of counts starts to become a significant factor. If it takes "too long" to do the counting, the capacitor can charge to either supply voltage depending on which direction it is integrating. This causes the wave shape shown in *Figure 15* to flatten out. This effectively limits the input range for all accuracy is lost once that waveform flattens out. In fact, this was the limiting factor on the accuracy in *Figure 16* as shown. Given the amount of time required for an increment of the counter for T_{REF} (or T_X), it was not possible to reach the 4096 counts required for 12 bit accuracy before the waveform flattened out. Decreasing the total count solves the problem at the expense of accuracy. It is therefore desirable to keep the loop time required for an increment as fast as possible. The code to implement *Figure 16* is shown in *Figure 17* and reflects that concern. The other way to solve the problem is to use a large value for R and C. This is the easiest solution and preserves accuracy. Its cost is increased conversion time.

Both the basic and modified dual slope schemes can be very accurate and are commonly used. They tend to be relatively slow. In many applications, however, speed is not a factor and these approaches can serve very well. There are various approaches to dual slope analog to digital conversion which try to improve speed and/or accuracy. These are usually multiple ramping schemes of one form or another. The heart of the approach is the basic scheme described above. It is not the purpose here to delve into all the possible ways that dual slope conversion may be accomplished. The control software is not significantly different regardless of which particular variation is used. The basic ramping control is the same as that indicated here.



TL/DD/6935-24

FIGURE 16. Modified Dual Slope Integration

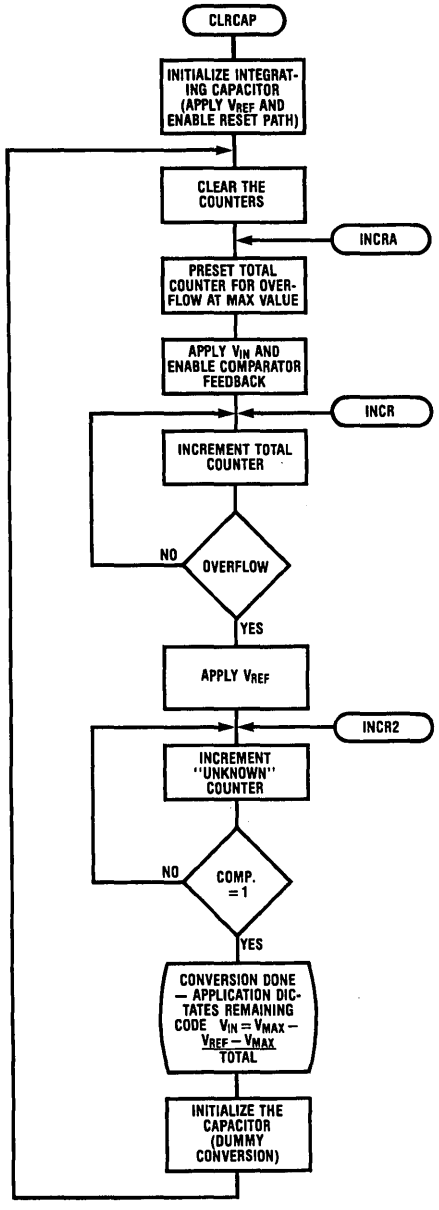
The number of components required to implement a dual slope scheme is not related to the desired accuracy. The approach is generally tolerant as to the op amps and comparators used as long as proper care is given to the initialization of the integrating capacitor.

Precise references are not required if a ratiometric system is all that is required. Cheaper switches can be safely used. The dual slope scheme controlled by a COPS microcontroller can be a very cost effective solution to an analog to digital conversion problem.

```
CLRCAP: OGI     1       ;APPLY VREF AND ENABLE RESET PATH
CLEAR2: LBI     2,11    ;NOW CLEAR THE COUNTER
        JSRP    CLEAR
  ;J,15=15,1,14=4 AND START AT 1,12 FOR COUNT = 3072
  ;J,15 =15 AND START AT 1,12 FOR COUNT = 4096
  ;J,15 = 14 AND START AT 1,12 FOR COUNT = 8192
  ;J,15 = 12 AND START AT 1,12 FOR COUNT = 16384
  ;FOLLOW SAME PATTERN FOR OTHER COUNTS
  ;
MEASUR: JSR     INCRA   ;RUN THRU THE INCREMENTS
        ;HAVE THE VALUE AT THIS POINT, DO WHAT THE APPLICATION
        ;REQUIRES--REMEMBER, TO CREATE REAL VALUE MUST MULTIPLY
        ;RESULT BY (VREF-VMAX)/TOTAL COUNT AND THEN SUBTRACT
        ;THAT RESULT FROM VMAX--DO IT IN DECIMAL OR BINARY, WHICHEVER
        ;IS BEST FOR THE APPLICATION
        LBI     1,11    ;MAKE SURE SPACE IS CLEARED
        JSRP    CLEAR
        LBI     2,11
        JSRP    CLEAR
        JSR     INCRB    ;FOR TEST-KEEP IT CLOSE
        LBI     1,11     ;MAKE SURE COUNTER IS CLEARED
        JSRP    CLEAR
        JP      CLEAR2
INCRA:  LBI     1,14
        STII    4        ;PRESET HERE FOR SMALLER COUNT
        STII    15       ;PRESET THE COUNTER FOR 4096
INCRA1: OGI     2        ;APPLY VIN AND ENABLE FEEDBACK
INCR:   LBI     1,12
        SC
BINAD1: CLRA
        ASC
        NOP
        XIS
        JP      BINAD1
        NOP              ;2 NOPS TO EQUALIZE TIMES
        NOP
        SKC
        JP      INCR
        OGI     0        ;DONE, NOW APPLY VREF
INCR2:  LBI     2,12     ;COUNT UNTIL COMPARATOR CHANGES
        SC
BINAD2: CLRA
        ASC
        NOP
        XIS
        JP      BINAD2   ;STRAIGHT LINE THE ADD FOR SPEED
        ININ             ;SAVE WORDS BY USING G
        AISC    8        ;SEE IF IN3=1
        JP      INCR2    ;IN1 IS 0, KEEP COUNTING
OUTPUT: OGI     1        ;CLEAR THE CAPACITOR, APPLY VREF
        RET
INCRB:  LBI     1,14     ;MAKE THE PASS FOR CAP INIT SHORT
        STII    7
        STII    15
        JP      INCRA1
```

TL/DD/6935-48

FIGURE 17A. Modified Dual Slope Code



TL/DD/6935-25

FIGURE 17B. Modified Dual Slope Flow Chart

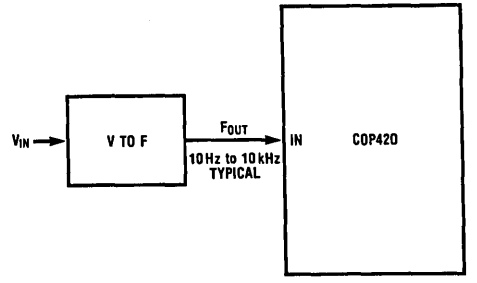
5.0 Voltage to Frequency Converters, VCO's

5.1 BASIC APPROACH

The basic idea of this scheme is simply to use the COP420 to measure the frequency output of a voltage to frequency converter or VCO. This frequency is in direct relation to the input voltage by the very nature of such devices. There are really only two limiting factors involved. First of all, the maximum frequency that can be measured is defined in the microcontroller by the amount of time required to test an input and increment a counter of the proper length. With the COP420 this upper limit is typically 10 to 15 kHz. The other limiting factor is simply the accuracy of the voltage to frequency converter or VCO. This accuracy will obviously affect the accuracy of the result.

Two basic implementations are possible and their code implementation is not significantly different. First, the number of pulses that occur within a given time period may be counted. This is straightforward and fairly simple to implement. The crucial factor is how long that given time period should be. To get the maximum accuracy from this implementation the time period should be one second. Such a time period would allow the distinction between the frequencies of 5000 Hz and 5001 Hz for example (assuming the V to F converter was that accurate or precise). Decreasing the amount of time will decrease the precision of the result. The alternate approach is to measure (by means of a counter) the amount of time between two successive pulses. This period measurement is only slightly more complicated than the pulse counting approach. The approach also makes it possible to do averaging of the measurement during conversion. This will smooth out any changes and add stability to the result. The time measurement technique is also faster than the pulse counting approach. Its accuracy is governed by how finely the time periods can be measured. The greater the count that can be achieved at the fastest input frequency — shortest period — the more accurate the result.

Figure 18 illustrates the basic concept. *Figure 19A* shows the flow charts and code implementation for both of the approaches discussed above. Note that whatever type of V to F converter is used, the code illustrated in *Figure 19A* is not significantly changed. In the code of *Figure 19A*, the interrupt is being used to test an input and thereby decreases the total time loop.



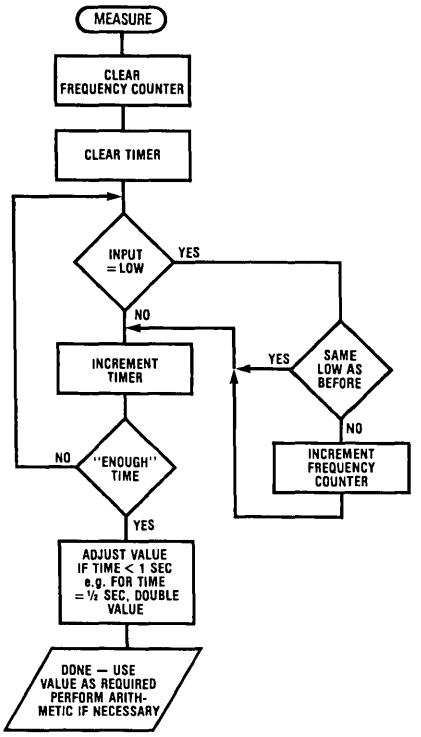
TL/DD/6935-26

FIGURE 18. V to F Converter — Basic Concept

```
MEASUR:          ;MEASURE BY COUNTING PULSES OF V TO F
;
         LEI     2          ;ENABLE INTERRUPT
         LBI     1,14       ;PRESET TIME FOR 122 COUNTS
         STII    5          ;APPROX ONE HALF SECOND
         STII    8
TIME:    SKT                ;USE INTERNAL TIMER TO FIND
         JP      TIME       ;THE 1/2 SECOND
BINPT1:  LBI     1,14       ;HAVE GOT IT, INCREMENT COUNTER
         SC
BINADD:  CLRA
         ASC
         NOP
         XIS
         JP      BINADD
         SKC                ;NOW SEE IF DONE
         JP      TIME       ;NO COUNTER OVERFLOW, CONTINUE
         LEI     0          ;DONE, DISABLE INTERRUPT
FIN:     ;AT THIS POINT HAVE THE VALUE--CONVERT IT TO DECIMAL OR
         ;SEND IT OUT OR PROCESS IT FURTHER, WHATEVER IS REQUIRED
         ;BY THE APPLICATION.  ARITHMETIC IS REQUIRED TO CREATE THE
         ;VOLTAGE VALUE, USUALLY A SIMPLE MULTIPLY
         ;MAY HAVE TO DOUBLE THE RESULT TO COMPENSATE LOOKING FOR
         ;ONLY 1/2 SECOND IN THIS CASE
         ;
         JP      MEASUR     ;DO IT OVER AGAIN
         .=X'0FF            ;SET ADDRESS TO 0FF FOR INTERRUPT
INTENT:  NOP                ;ADDRESS 0FF MUST BE NOP FOR INTERRUPT
INTRPT:  LBI     2,12       ;DO ADD OF THE VALUE FOR FREQ CNT
         SC
INTRT:   CLRA               ;STRAIGHT LINE THE CODE FOR SPEED
         ASC
         NOP
         XIS
         CLRA
         ASC
         NOP
         XIS
         CLRA
         ASC
         NOP
         XIS
         CLRA
         ASC
         NOP
         X
         LEI     2          ;ENABLE THE INTERRUPT AGAIN
         RET
```

TL/DD/6935-49

FIGURE 19A. V to F by Counting Pulses



TL/DD/6935-27

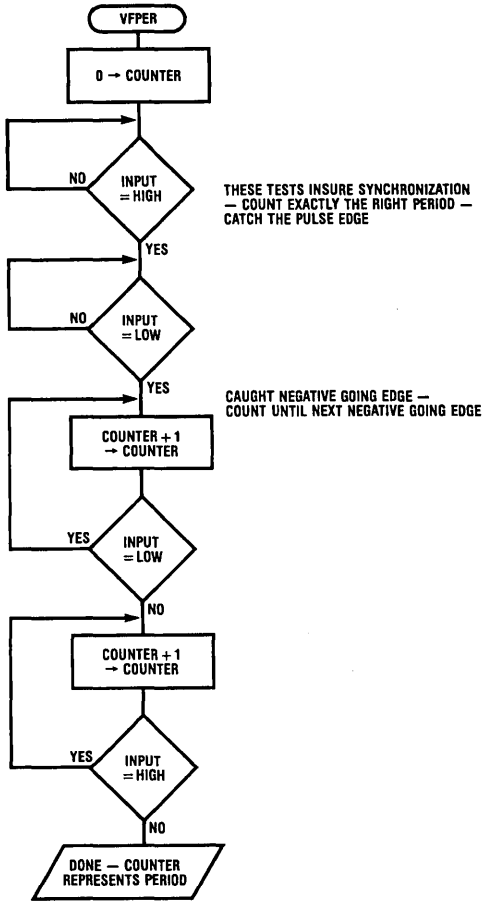
FIGURE 19B. V to F by Counting Pulses

```
         ;USE INTERRUPT FOR CATCHING THE PULSE EDGE

VFPER:   LBI     0,12       ;CLEAR COUNTER SPACE AND FLAG
         STII    0
         STII    0
         STII    0
         STII    0
         LBI     0,12
         LEI     2          ;NOW ENABLE THE INTERRUPT
WAIT:    SC                 ;DUMMY WAIT LOOP, WAITING FOR SIGNAL TO
         LBI     0,12       ;INTERRUPT THE CONTROLLER
         JP      WAIT
         .=X'0FF            ;SET ADDRESS TO OFF--INTERRUPT ENTRY POINT
INTENT:  NOP                ;REQUIRED FOR INTERRUPT ENTRY
COUNT:   LBI     0,12       ;NOW CHECKING TO SEE IF SECOND INTERRUPT
         SKMBZ   0          ;I.E., ARE WE DONE?
         JP      DONE
         SMB     0          ;SET BIT FOR NEXT INTERRUPT
         LEI     2          ;ENABLE INTERRUPT AGAIN
PLUS1:   LBI     0,13       ;NOW START COUNTING
         SC
         CLRA               ;STRAIGHT LINE THE CODE FOR SPEED
         ASC
         NOP
         XIS
         CLRA
         ASC
         NOP
         XIS
         CLRA
         ASC
         NOP
         X
         JP      PLUS1
DONE:    ;FINISHED WHEN GET HERE--THE COUNT REPRESENTS THE PERIOD
         ;WITH ABOVE CODE, THE ACTUAL PERIOD IS THE COUNT MULTIPLIED
         ;BY 15(THE NUMBER OF WORDS TO INCREMENT BY 1) PLUS AN OVERHEAD
         ;OF 9 CYCLE TIMES = 24 CYCLE TIMES.  AT 4us THIS IS 96 us
         ;OR A FREQUENCY OF JUST OVER 10KHz.  MAX COUNT HERE IS 4095.
         ;THIS GIVES A MAXIMUM PERIOD = 61434 CYCLE TIMES(=245.736ms AT
         ;4us).  THIS CORRESPONDS TO A FREQUENCY OF JUST OVER 4Hz
         ;NOTE, THIS IS 12 BIT RESOLUTION
```

TL/DD/6935-50

FIGURE 19C. A to D with VF Converter/VCO by Measuring Period



TL/DD/6935–28

FIGURE 19D. V to F—Measure Period

5.2 THE LM131/LM231/LM331

The LM131 is a standard product voltage to frequency converter with a linear relationship between the input voltage and the resultant frequency. The reader should refer to the data sheet for the LM131 for further information on the device itself and precautions that should be taken when using the device. *Figure 20* is the basic circuit for using the LM131. *Figure 21* represents improvements that increase the accuracy (by increasing the linearity) of the result. Note that these circuits have been taken from the data sheet of the LM131 and the user is referred there for a further discussion of their individual characteristics. With the LM131 the frequency output is given by the relationship:

$$F_{OUT} = (V_{IN}/2.09)\,(1/R_TC_T)\,(R_S/RL)$$

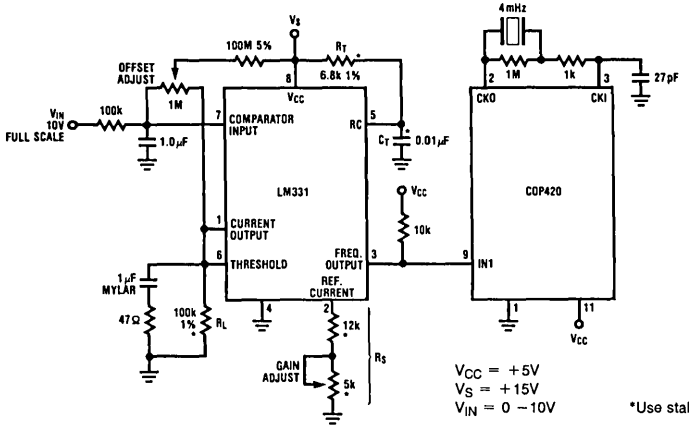
It is clear from the expression above that the accuracy of the result depends upon the accuracy of the external components. The circuit may be calibrated by means of a variable resistance in the R_S term (a gain adjust) and an offset adjust. The offset adjust is optional but its inclusion in the circuit will allow maximum accuracy to be obtained. The standard calibration procedure is to trim the gain adjust (R_S) until the output frequency is correct near full scale. Then set the input to 0.01 or 0.001 of full scale and trim the offset adjust to get F_{OUT} to be correct at 0.01 or 0.001 of full scale. With that calibration, the circuit of *Figure 20* is accurate to within ±0.03% typical and ±0.14% maximum. The circuit of *Figure 21* attains the spec limit accuracy of ±0.01%.

5.3 VOLTAGE CONTROLLED OSCILLATORS (VCO's)

A VCO is simply another form of voltage to frequency converter. It is an oscillator whose oscillation frequency is dependent upon the input voltage. Numerous designs for VCO's exist and the reader should refer to the data sheets and application notes for various op-amps and VCO devices. The code in *Figure 19* is still applicable if a VCO is used. The only possible difficulty that might be encountered is if the relationship between frequency and input voltage is non-linear. This does not affect the basic code but would affect the processing to create the final result. A sample circuit, taken from the data sheet of the LM358, is shown in *Figure 22*. The accuracy of the VCO is the controlling factor.

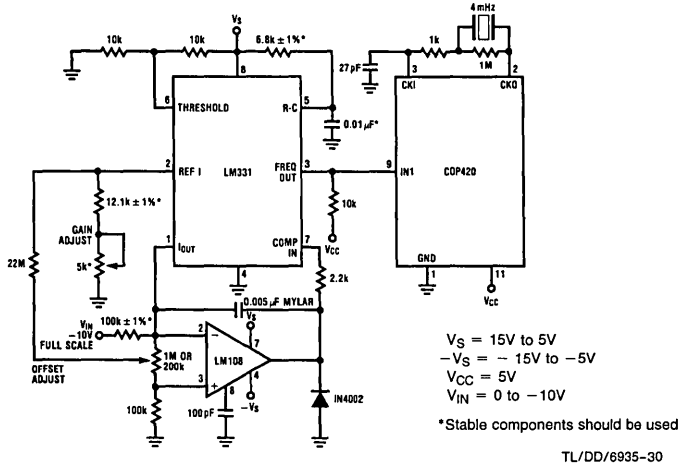
5.4 A COMBINED APPROACH

Elements of the period measurement and pulse counting techniques can be combined to produce a system with the advantages of both schemes and with few problems. Such a system is only slightly more complicated in terms of its software implementation than the approaches mentioned above. Note that in a microcontroller driven system, no additional hardware beyond the voltage to frequency converter is required to implement this approach. Basically, the microcontroller establishes a viewing window during which time the microcontroller is both measuring time and counting pulses. The result can be very precise if two conditions are met. First, when the microcontroller determines that it needs the conversion information, the microcontroller does not begin counting time or pulses until the first pulse is received from the VFC (first pulse after the microcontroller "ready"). Note, the COPS microcontroller could provide a "start conversion" pulse to enable the VFC if such an arrangement were desirable. The time would be counted for a fixed period and the number of pulses would be counted. After the fixed period of time the controller would wait for the next pulse from the VFC and continue to count time until that pulse is received. The ratio of the total time to the number of pulse is a very precise result provided that all the system times are slow enough that the microcontroller can do its job. The speed limits mentioned previously apply here. It is clear that the total time is not fixed. It is some basic time period plus some variable time. This is a little more complicated than simply using a fixed time, but it allows greater accuracies to be achieved. Also, the approach takes approximately the same amount of time for all conversions. It is also faster than the simple pulse counting scheme.



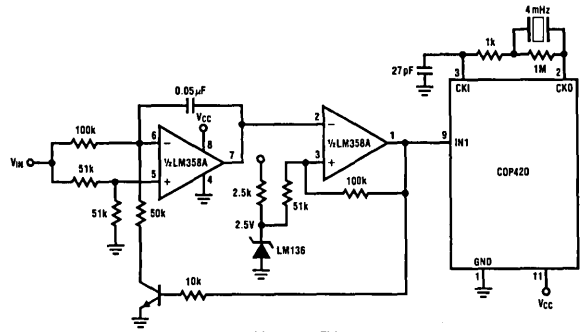
TL/DD/6935-29

FIGURE 20. Basic LM331 Connection



TL/DD/6935-30

FIGURE 21. A to D with Precision Voltage to Frequency Converter



V_{CC} = +5V
V_{IN} = 0–5V

TL/DD/6935-31

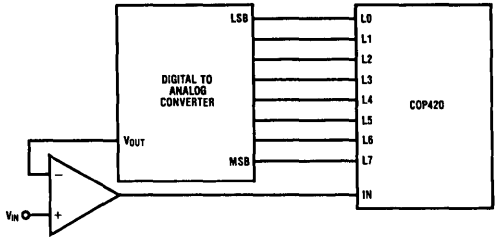
FIGURE 22. A to D with VCO

6.0 Successive Approximation

6.1 BASIC APPROACH

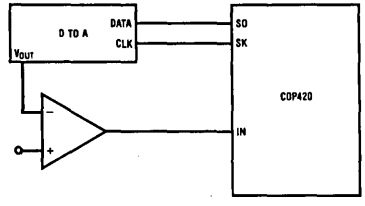
The successive approximation technique is one of the more standard approaches in analog to digital conversion. It requires a counter or register (here provided by the COP420), a digital to analog converter, and a comparator. *Figure 23A/B* illustrates the basic idea with the COP420. In the most basic scheme, the counter is reset to zero and then incremented until the voltage from the digital to analog converter is equal to the input voltage. The equality is determined by means of the comparator. *Figure 24B* illustrates the flow chart and code for this most basic approach. The preferred approach is illustrated in *Figure 25A/B*. This is the standard binary search method. The counter or register is set at the midpoint and the "delta" value set at one half the midpoint. The "delta" value is added or subtracted from the initial guess depending on the output of the comparator. The "delta" value is divided by 2 before the next increment or decrement. The method repeats until the desired resolution is achieved. While this approach is somewhat more complicated than the basic approach it has the advantage of always taking the same amount of time for the conversion regardless of the value of the input voltage. The conversion time for the basic approach increases with the input voltage. The preferred approach is almost always faster than the basic approach. The basic approach is faster only for those voltages near zero where it has only a few increments to perform.

The accuracy of the approach is governed by the accuracy of the digital to analog converter and the comparator. Thus, the result can be as accurate as one desires depending on the choice of those components. Digital to analog converters of various accuracies are readily available as standard parts. Their cost is usually in direct relation to their accuracy. The reader should refer to the National Semiconductor Data Acquisition Handbook for some possible candidates for digital to analog converters. It is not the purpose here to compare those parts. The COPS interface to these parts is generally straightforward and follows the basic schematics shown in *Figure 23*. The user should take note and make sure the input and output ports of the converter are compatible — in terms of voltages and currents — with the COPS device. This is generally not a problem as most of the parts are TTL compatible on input and output. The precautions and restrictions as to the use of any given device are governed by that device and are indicated in the respective data sheets.



TL/DD/6935-32

FIGURE 23A. Basic Parallel Implementation



TL/DD/6935-33

FIGURE 23B. Basic Serial Implementation

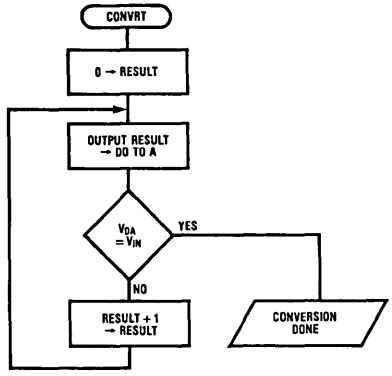
```
                ;8 BIT SUCCESSIVE APPROXIMATION--BASIC SCHEME
                ;COMPARATOR INPUT TO COP = IN3
                ;OUTPUTS TO D TO A ARE L7 THRU L0 WITH L7 = MSB, L0 = LSB

CONVRT: LBI     2,14    ;SET THE RESULT VALUE TO ZERO
        STII    0
        STII    0
        LEI     4       ;ENABLE THE L PORT AS OUTPUTS
        JP      OUTPUT
INCR:   SC              ;ROUTINE FOR INCREMENTING THE RESULT VALUE
PLUS1:  CLRA
        LBI     2,14
        ASC
        NOP
        XIS
        JP      PLUS1
OUTPUT: LBI     2,15    ;SEND THE RESULT VALUE, STORED IN 2,15-2,14 TO
        LD              ;Q AND THEREBY OUT THROUGH L
        XDS
        CAMQ
        JSR     DELAY   ;THIS IS ANY CONVENIENT ROUTINE TO MAKE SURE
                        ;THAT THE COP DOES NOT TEST THE COMPARATOR UNTIL
                        ;THE D TO A CONVERTER HAS HAD ENOUGH TIME TO DO
                        ;THE CONVERSION--THE AMOUNT OF TIME REQUIRED
                        ;IS CLEARLY DEPENDANT UPON THE D TO A CONVERTER
                        ;USED
        ININ            ;NOW READ THE COMPARATOR INPUT TO COP
        AISC    8       ;COULD SAVE A WORD IF USE Q LINE AS INPUT
        JP      INCR    ;INPUT VOLTAGE STILL > CONVERTED ANALOG VOLTAGE

;CONVERSION DONE AT THIS POINT--THE COMPARATOR HAS CHANGED STATE
;HENCE, CONVERTED ANALOG VOLTAGE > INPUT VOLTAGE--SO STOP
```

TL/DD/6935-51

FIGURE 24A. Code for Basic Approach of Successive Approximation



TL/DD/6935-34

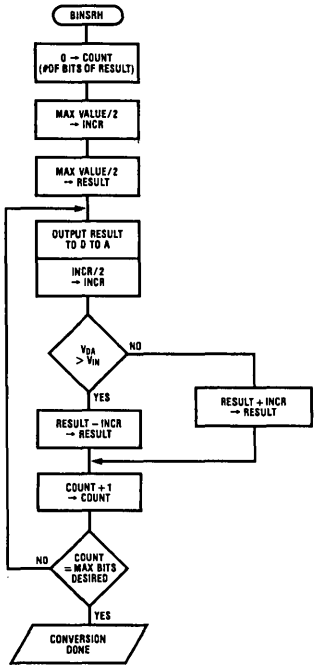
FIGURE 24B. Basic Approach, Successive Approximation

```
        ;8 BIT BINARY SEARCH SUCCESSIVE APPROXIMATION
        ;INPUT TO COP IS IN3,L BUS IS OUTPUT TO D TO A,L7=MSB,L0=LSB
        ;COMPARATOR=0 WHEN D TO A VOLTAGE > VIN, OTHERWISE = 1

BINSRH: LBI     3,14    ;SET INCREMENT = MAX VALUE/2(WILL BECOME
        STII    0       ;MAX VALUE/4 BEFORE FIRST USE)
        STII    8
        LBI     2,14    ;SET INITIAL VALUE OF RESULT TO MAX VALUE/2
        STII    0
        STII    8
        LEI     4       ;ENABLE THE L BUS AS OUTPUTS
        LBI     1,15    ;NOW SET UP THE BIT COUNTER-OVERFLOW WHEN 8 BITS
        CLRA
        AISC    9       ;DO IT THIS WAY FOR COMPATIBILITY WITH INCREMENT
OUTPUT: X       3       ;SAVE THE BIT COUNTER VALUE AND POINT TO RESULT
        LD
        XDS             ;SEND THE RESULT TO Q AND HENCE TO L
        CAMQ
DIVIDE: LBI     3,15    ;DIVIDE THE INCREMENT VALUE BY 2, CAN BE DONE
DIVA:   LD              ;IN SEVERAL WAYS SINCE THIS IS A VERY SPECIAL
        AISC    8       ;PURPOSE DIVIDE FUNCTION
        JP      DIV1    ;ALSO,DO THE DIVIDE HERE TO GIVE THE D TO A TIME
        STII    4       ;TO DO THE DIGITAL TO ANALOG CONVERSION
        JP      TEST
DIV1:   AISC    4
        JP      DIV2
        STII    2
        JP      TEST
DIV2:   AISC    2
        JP      DIV3
        STII    1
        JP      TEST
DIV3:   LBI     3,14
        AISC    1
        JP      DIVA
        STII    8
        STII    0
        ;DEPENDING ON THE D TO A USED, MAY NEED MORE DELAY HERE
        ;MUST BE SURE THE RESULT IS STEADY BEFORE TEST THE COMPARATOR
TEST:   LBI     3,14
        ININ
        AISC    8       ;COULD SAVE A WORD IF USED Q LINE AS INPUT
        JP      INCR
DECR:   SC              ;INPUT LESS THAN D TO A CONVERTED VOLTAGE
SUB:    LD      1       ;SUBTRACT THE INCREMENT VALUE FROM RESULT
        CASC
        NOP
        XIS     1
        JP      SUB
        JP      BITPL1
INCR:   RC              ;INPUT > D TO A CONVERTED VOLTAGE
ADD:    LD      1       ;ADD THE INCREMENT VALUE TO RESULT VALUE
        ASC
        NOP
        XIS     1
        JP      ADD
BITPL1: LBI     1,15    ;NOW INCREMENT BIT COUNTER TO SEE IF DONE
        LD
        AISC    1
        JP      OUTPUT
        ;CONVERSION DONE AT THIS POINT
```

TL/DD/6935-52

FIGURE 25A. Binary Search Successive Approximation Code



TL/DD/6935-35

FIGURE 25B. Binary Search Successive Approximation Flow Chart

6.2 SOME COMMENTS ON RESISTOR LADDERS

If the user does not wish to use one of the standard digital to analog converters, he can always build one of his own. One of the most standard methods of doing so is to use a resistor ladder network of some form. *Figure 26* illustrates the basic forms of binary ladders for digital to analog converters. The figures also show the transition from the basic binary weighted ladder in *Figure 26A* to the standard R-2R ladder *Figure 26C*.

Consider *Figure 26A*. The choice of the terminating resistor is made by hypothesizing that the ladder were to go on ad infinitum. It can then be shown that the equivalent resistance at point X in that figure would be equal to 128R, the same value as the resistor to the least significant bit output. This fact is used to create the intermediate ladder of *Figure 26B*. This step is done because it is usually undesirable to have to find the multitude of resistor values required in the basic binary ladder. Thus, the modification in *Figure 26B* significantly reduces the number of resistor values required. As stated earlier, the resistance looking down the ladder at point X in *Figure 2* is equal to the resistor connected to the binary output at that point; here the value is 2R. Remembering the objective is to minimize the number of different values required, if we simply use the same R-2R arrangement as before with a termination of 2R we get an effective resistance at point Y of *Figure 26B* or 0.5R. This means that a serial resistance of 1.5R is required to maintain the integrity of the ladder. If we carry this on through 8 bits, the circuit of *Figure 26B* results. From this it is only a small step to create the standard R-2R network. The analysis is the same as done previously.

There is absolutely no restriction that the ladders must be binary. A ladder for any type of code can be constructed with the same techniques. Ladders comparable to *Figures 26A* and *26B* are shown in *Figure 27* for a standard 8421 BCD code. With the BCD code, the input must be considered in groups of digits with four bits creating one digit. This is the direct analog of 1 binary digit per unit. We need four inputs to create one decimal digit. Thus the resistor values in each decimal digit are 10 times the values in the previous decimal digit just as the resistor value for each successive binary digit was twice the value for the preceding binary digit. Note that this analysis can be easily extended to any code. The termination resistance is calculated in the same manner—assume the decimal digit groupings extend out to infinity. It can be shown that the resistance of the ladder at point X in *Figure 27A* is 480R. Thus *Figure 27A* represents the basic 8241 BCD ladder for three digit BCD number. This termination resistance will vary with where it is placed. Basically this resistance is equal to nine times (for a decimal ladder) the parallel resistance of the last digit implemented. (This relation can be shown mathematically if one desired, the multiplier is a function of the type of ladder used—multiplier = 1 for binary systems, 9 for decimal systems, etc.) Thus the termination resistance would be 48R if the network were terminated after the 2nd digit and 4.8R if the network were terminated after the 1st digit implemented. In

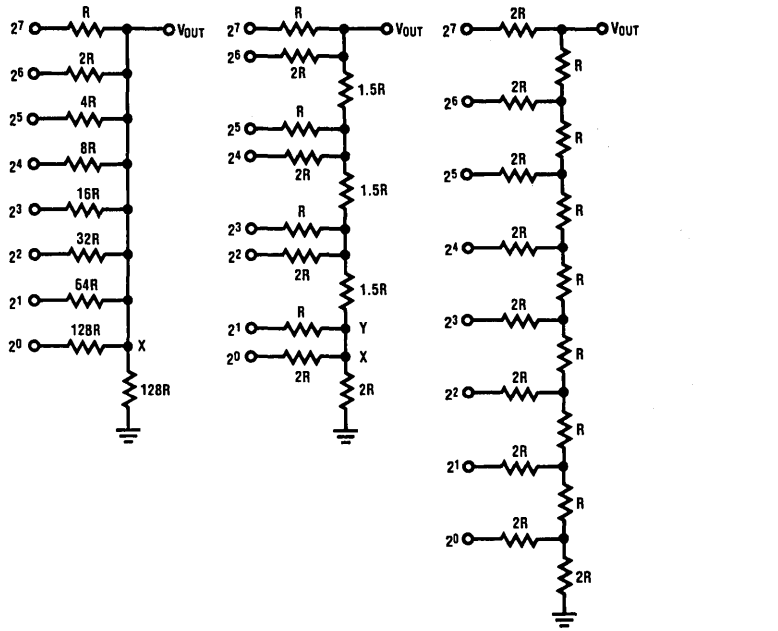


FIGURE 26. Binary Ladders

Figure 27B we are attempting to use only the resistor values for one decimal digit. This means that the last terminating resistor must be a 4.8R by the analysis above. Thus at point X in *Figure 27B* we must have an equivalent of resistance of 4.8R. The equivalent resistance at point Y of *Figure 27B*, looking down from the ladder, is 0.48R. Thus the other series resistance must be 4.32 R (4.8R−0.48R). Thus the network of *Figure 27B* results.

Generally, ladders can be very effective tools when understood and used properly. They can be significantly more involved than indicated here. There are a number of texts and articles that cover the subject very nicely and the reader is referred to them if more information on ladder design, the use of ladders, and advanced techniques with ladders is desired.

One final note is of some interest. The ladders may be readily constructed for any type of code to create the analog voltage. Note that there is no restriction that the code, or the ladder network, be linear. Thus, effective use of ladder networks may significantly reduce system difficulties and complexities caused by the fact that the analog to digital conversion is being performed on a voltage source that changes nonlinearly, for example a thermistor temperature probe. By using the properly designed ladder network, the nonlinearity can effectively be eliminated from consideration in the code implementation of the analog to digital conversion.

The accuracy of ladders is a direct function of the accuracy of the resistors and the accuracy of the voltage source inputs. This is obvious since the analog voltage is in fact created by means of equivalent voltage dividers created when the various inputs are on or off. It is also essential that the ladder sources be the precise same value at all inputs to the ladder network. If this is not the case, errors will be introduced. In addition, the output impedance of the voltage source should be as small as possible. The success of the ladder scheme depends on the ratios of the resistance values. Inaccuracies are introduced if those ratios are disturbed. Some possible implementations of the successive approximation approach with a ladder network used for the digital to analog conversion are indicated in *Figure 28*.

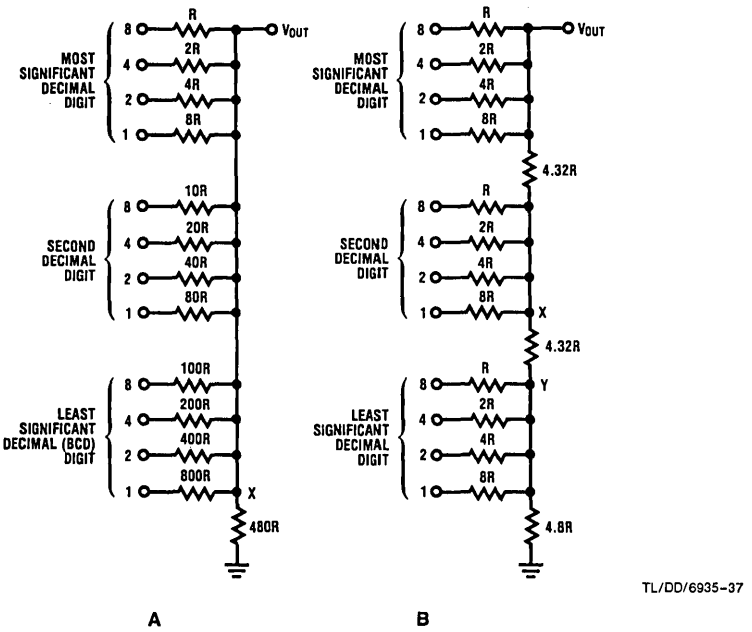
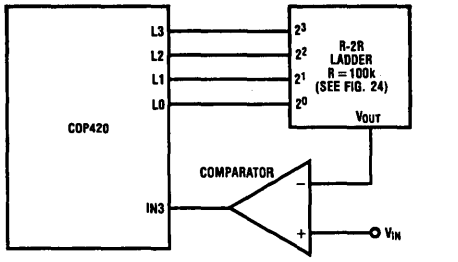


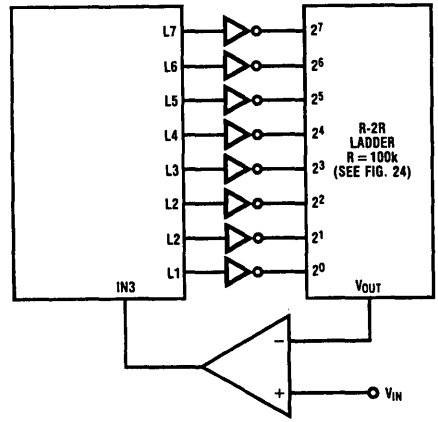
FIGURE 27. 8421 BCD Ladders

Note that these are functional diagrams. Feedback or hysteresis for comparator stabilization are not shown. The reader should be aware that his particular application may require that these factors be considered. *Figure 28A* is the simplest scheme and also the least accurate. With little or no load, the high output level of the L buffer should be very close to V_{CC} and the low level close to ground. Also the output impedance of the buffers must be considered. Therefore, rather large resistor values are used—both to keep the load very small and to dwarf the effect of the output impedance. With the configuration in *Figure 28A*, four bit accuracy is about the best that can be achieved. By being extremely careful and using measured values, an additional bit of accuracy may be obtained but care must be used. However, the schematic of *Figure 28A* is very simple. *Figure 28B* represents the next step of improvement. Here we have placed CMOS buffers in the network. This eliminates the output impedance and reduces the level problems of the circuit of *Figure 28A*. The CMOS buffer will swing rail to rail, or nearly so. The accuracy of V_{CC} and the resistor network is then



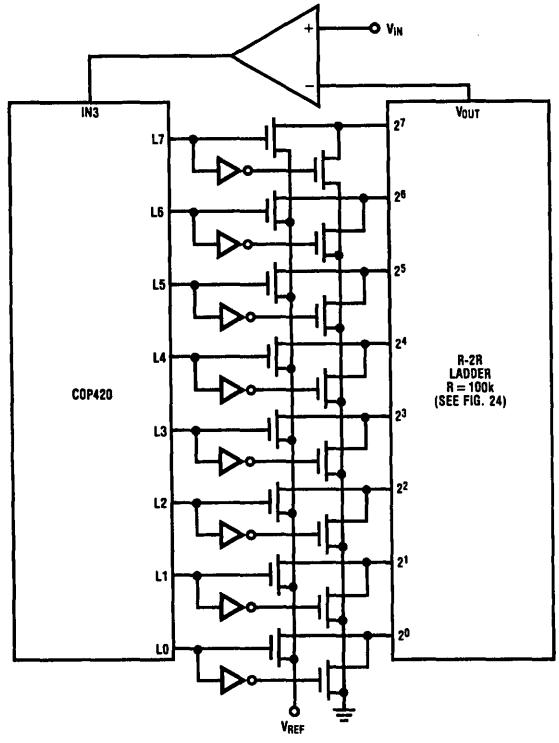
TL/DD/6935-38

A



TL/DD/6935-39

B



TL/DD/6935-40

C

FIGURE 28. Interfaces to Ladder Networks

controlling. Using 1% resistors and holding V_{CC} constant, the user should be able to achieve 7 to 8 bit accuracy without much difficulty. Remember, however, that V_{CC} is one of the controlling factors. If V_{CC} is ±5%, there is no point in using 1% resistors since the V_{CC} tolerance swamps their effect. *Figure 28C* is the final and most accurate approach. Naturally enough, it is the most expensive. However, one can get as accurate as one desires. Here, an accurate reference is required. That reference is switched into the network by means of the analog switch. Alternately, ground may be connected to the input. Now the user need only consider the accuracy of the reference and the accuracy of the resistors. However, the on impedance of the switches must be considered. It is necessary to make this on impedance as low as possible so as not to alter the effective resistor values.

7.0 "Offboard" Techniques

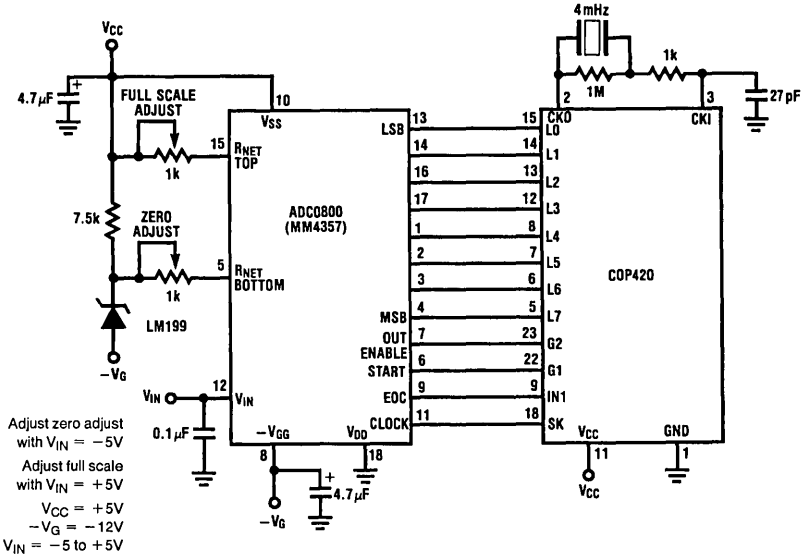
7.1 GENERAL COMMENTS

This section is devoted to a few illustrations of interfacing the COP420 to standard, stand alone analog to digital converters. These standard converters are used as peripherals to the COPS device. Whenever the microcontroller requires a new reading of some analog voltage, it simply initiates a read of the peripheral analog to digital converter. As a result, the accuracies and restrictions in using the converters are governed by those devices and not by the COPS device. These techniques are generally applicable to other A to D converters not mentioned here and the user should not have difficulty in applying these principles to other devices. It should be pointed out that in almost every instance, the choice of COP420 inputs and outputs is arbitrary. Obviously, when there is an 8-bit bus it is natural, and most efficient, to use the L port to interface to the bus. Generally, the G lines have been used as outputs rather than the D lines simply because the G lines are, in many instances, somewhat easier to control. The choice of input line is also free. If the interrupt is not otherwise being used, it may be possible to utilize this feature of IN1 for reading a return signal from the converter. However, this is by no means required. If there is a serial interface it is clearly more efficient to use the serial port of the COP420 as the interface. If a clock is required, SK is the natural choice.

7.2 ADC0800 INTERFACE

The ADC0800 is an 8-bit analog to digital converter with an 8-bit parallel output port with complementary outputs. The ADC0800 requires a clock and a start convert pulse. It generates an end of conversion signal. There is an output enable which turns the outputs on in order to read the 8-bit result.

The reader is referred to the data sheet for the ADC0800 for more information on the device. The circuit of *Figure 29* illustrates the basic implementation of a system with the ADC0800. The interface to the COP420 is straightforward. The appropriate timing restrictions on the control signals are easily met by the microcontroller.



TL/DD/6935-41

FIGURE 29. Simple A/D with ADC0800

Figure 30 is the flow chart and code required to do the interfacing. As can be seen, the overhead in the COP420 device is very small. The choice of inputs and outputs is arbitrary. The only pin that is more or less restricted is the use of SK as the clock for the converter. SK is clearly the output to use for that function as, when properly enabled, it provides pulses at the instruction cycle rate.

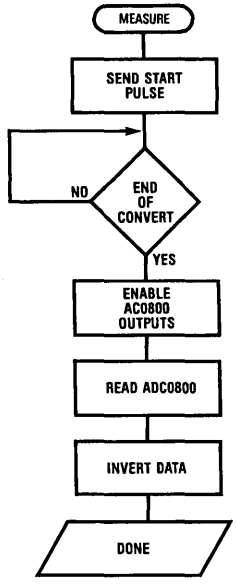
7.3 ADC0801/2/3/4 INTERFACE

The ADC0801 family of analog to digital converters is very easy to interface and is generally a very useful offboard converter. The interface is not significantly different from that of the ADC0800, but the ADC0801 famliy are a much better device. The four control signals are somewhat different, although there are still four control lines. Here we have a chip select, a read, a write, and an interrupt signal. All are negative going signals. Start conversion is the ANDing of chip select and write. Output enable is the ANDing of chip select and read. The interrupt output is an end convert signal of sorts. The device may be clocked externally or an RC may be connected to it and it will generate its own clock for the conversion. In addition the device has differential inputs

```
MEASUR: LEI     0       ;FLOAT THE L LINES
        SC
START2: CLRA            ;MAKE SURE SO STAYS ZERO
        XAS             ;MAKE SURE SK STAYS CLOCK
        OGI     2       ;SEND START PULSE
        OGI     0
        LBI     2,13
READ11: ININ
        AISC    14      ;WAIT FOR EOC SIGNAL
        JP      READ11
        OGI     4       ;HAVE EOC, ENABLE OUTPUTS
        INL             ;READ THE L LINES
        X
        COMP            ;CREATE PROPER POLARITY
        XDS
        COMP
        X
        OGI     0       ;DISABLE ADC0800 OUTPUT
        ;HAVE THE RESULT AT THIS POINT--USE IT IN WHATEVER
        ;MANNER IS REQUIRED BY THE APPLICATION
        LBI     2,10
        JSRP    CLRR
        JP      MEASUR
```

TL/DD/6935-53

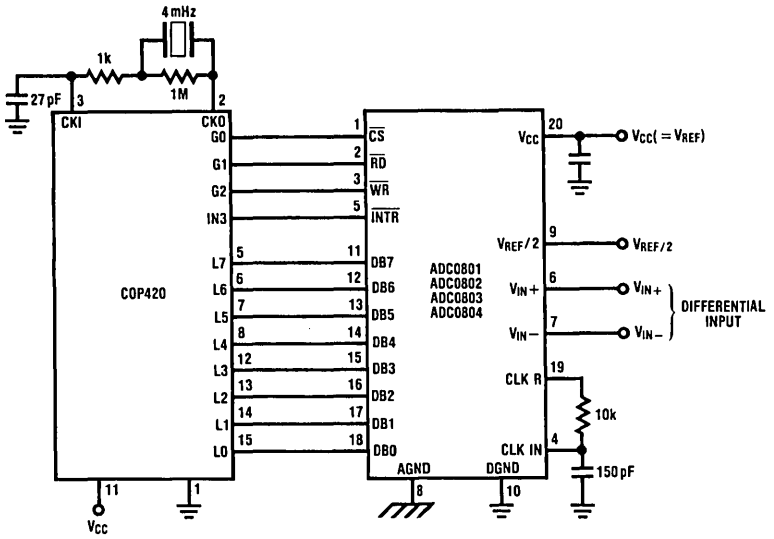
FIGURE 30A. A to D with ADC0800



TL/DD/6935-42

FIGURE 30B. ADC0800 Interface Flow

which allow the 8-bit conversion to be performed over a given window or range of input voltages. The reader should refer to the ADC0801 family data sheet for more information. *Figure 31* indicates a basic interface of the ADC0801 family to the COP420. Again, the interface is simple and straightforward. The code required to interface to the device is minimal. *Figure 32* illustrates the flow chart and code required to do the interface.



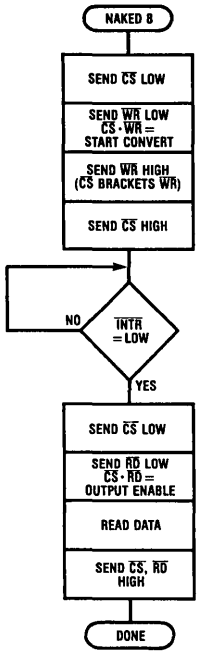
TL/DD/6935-43

FIGURE 31. COP420—ADC0801 Family Interface

```
        ;INTERFACE TO NAKED 8
        ;
NAKED:  OGI     15      ;SET ALL G LINES HIGH(USUALLY DONE AT
                        ;POWER UP
        LEI     0       ;TRI STATE THE L LINES FOR READING
LOOP:   OGI     14      ;SEND CHIP SELECT LOW(CS BRACKETS OTHER SIGNAL)
        OGI     10      ;CS LOW AND WR LOW = START CONVERSION
        OGI     14      ;RAISE WR
        OGI     15      ;RAISE CS, NAKED 8 IS NOW CONVERTING
LOOP2:  ININ            ;WAIT FOR THE INTR SIGNAL--COULD SAVE THIS TES
        AISC    8       ;IF USED IN1 AND THE INTERRUPT FEATURE OF COP4
        JP      READ    ;INTR IS LOW, DATA IS READY
        JP      LOOP2
READ:   LBI     0,0     ;SET UP RAM LOCATION FOR READ
        OGI     14      ;SEND CS
        OGI     12      ;SEND CS AND READ = OUTPUT ENABLE
        NOP             ;WAIT-NEED WAIT ONLY 125NS, BUT 1 CYCLE IS MIN
                        ;TIME WE CAN WAIT
        INL             ;READ THE L LINES
        OGI     15      ;TURN OFF THE NAKED 8--CS AND RD HIGH
        ;
        ;DONE AT THIS POINT, DO WHATEVER IS REQUIRED  WITH THE RESULT
        ;
```

TL/DD/6935-54

FIGURE 32A. COP420/ADC0801 Family Sample Interface Code



TL/DD/6935–44

FIGURE 32B. COP420/ADC0801 Family Interface Flow

8.0 Conclusion

Several analog to digital techniques using the COPS family have been presented. These are by no means the only techniques possible. The user is limited only by his imagination and whatever parts he can find. The COPS family of parts is extremely versatile and can readily be used to perform the analog to digital conversion in almost any method. Generally, those techniques where the COPS device is doing the counting or timekeeping are slow. However, those techniques are generally slow inherently. The fastest methods are those where the conversion is being done offboard and the COPS device is merely reading the result of the conversion when required. Also, an attempt has been made to illustrate the lower cost techniques of analog to digital conversion. This, by itself, restricts most of the techniques described to about 8-bits accuracy. As was mentioned several times, the greater the accuracy that is desired the more accurate the external circuits must be. Ten and twelve-bit accuracies, and more, require references that are accurate. These get very expensive very rapidly. There is nothing inherent in the COPS devices that prevents them from being used in accurate systems. The precautions are to be taken in the system regardless of the microcontroller. The only problem is that, in those accurate systems where the COPS device is doing the timekeeping and counting, this increased accuracy is paid for by increased time to perform the conversion.

Several devices have been used in conjunctions with the COPS device in the previous sections. It is again recommended that the user refer to the specific data sheets of those devices when using any of those circuits. It must again be mentioned that the standard precautions when dealing with analog signals and circuits must be taken. These are described in the National Semiconductor Linear Applications Handbook and in the data sheets for the various linear devices. These precautions are especially significant when greater accuracy is desired.

The COPS family of microcontrollers has shown itself to be very versatile and powerful when used to perform analog to digital conversions. Most techniques are code efficient and the microcontroller itself is almost never the limiting factor. It is hoped that this document will provide some guidance when it is necessary to perform analog to digital conversion in a COPS system.

9.0 References

1. "Digital Voltmeters and the MM5330", National Semiconductor Application Note AN-155.
2. Walker, Monty, "Exploit Ladder Network Design Potential". Part One of two part article on ladder networks. Magazine and date unknown.
3. Wyland, David C., "VFC's give your ADC design high resolution and wide range". *EDN*, Feb. 5, 1978.
4. Redfern, Thomas P., "Pulse Modulation A/D Converter" *Society of Automotive Engineers Congress and Exposition Technical paper #780435*, March 1978.
5. National Semiconductor Linear Applications Handbook, 1978.
6. National Semiconductor Linear Databook, 1980.
7. National Semiconductor Data Acquisition Handbook, 1978.

The COP444L Evaluation Device 444L-EVAL

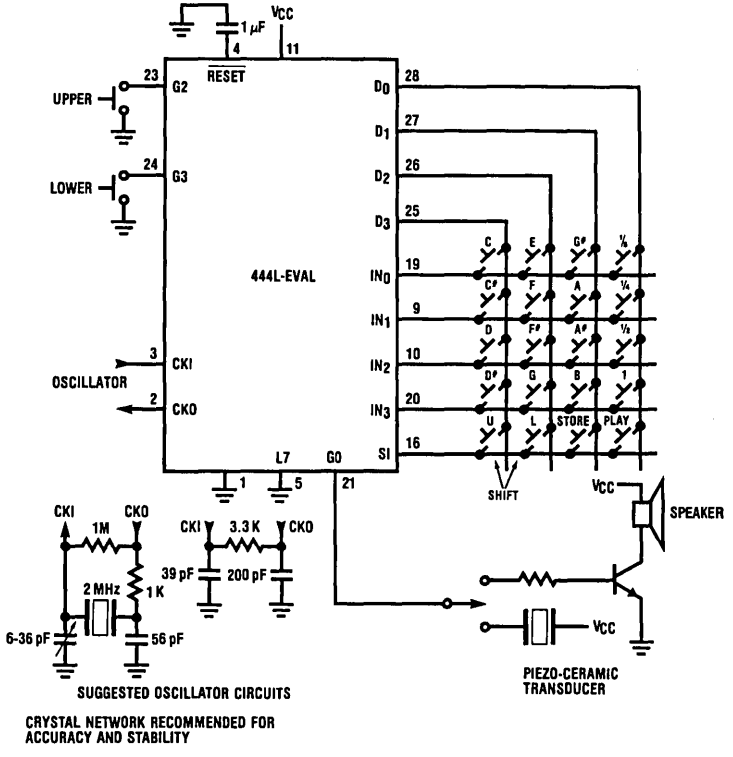
National Semiconductor
COP Note 4
Leonard A. Distaso

The 444L-EVAL is a preprogrammed COP444L intended to demonstrate operating characteristics and facilitate user familiarization and evaluation of the COP444L and the COPS™ family in general.

The 444L-EVAL has two mutually exclusive operating modes: an up/down counter/timer or a simple music synthesizer. The state of pin L7 at power up determines the operating mode.

1.0 THE 444L-EVAL AS A SIMPLE MUSIC SYNTHESIZER

Figure 1 indicates the connection of the 444L-EVAL as a simple music synthesizer. As the diagram indicates, the connections required for operation are minimal. The oscillator may be a crystal circuit using CKI and CKO; an external oscillator to CKI; or an RC network using CKI and CKO. As should be expected, the crystal circuit provides the greatest frequency stability and precision. The RC network will provide an acceptable oscillation frequency but that frequency will be neither precise nor stable over temperature and voltage. The external oscillator, of course, is as good as its source. The frequencies for the various notes and delay times are set up assuming that the oscillator frequency is 2 MHz. Three modes of operation are available in the music synthesizer mode: play a note; play one of four stored tunes; or record a tune for subsequent replay.



TL/DD/6937–1

FIGURE 1. 444L-EVAL as Simple Music Synthesizer

1.A. PLAY A NOTE

Twelve keys, representing the twelve notes in one octave, are labeled "C" through "B". Depressing a key causes a square wave of the corresponding frequency to output at GO. The user may drive a piezo-ceramic transducer directly with this signal. With the appropriate buffering, the user may use this signal to drive anything he wishes. A simple transistor driver is sufficient to drive a small speaker. The user can be as simple or as complex as he desires at this point—e.g. he can do some wave shaping, add an audio amplifier, and drive a high quality speaker.

The 444L-EVAL has a range of two and one-half octaves: the basic octave on the keyboard (which is middle C and the 11 notes above it in the chromatic scale), one full octave above the basic octave and one-half octave below the basic octave. The notes in the basic octave are played by depressing the appropriate key (one key at a time—the keyboard has no rollover provisions). A note in the upper octave is played by first depressing and releasing the U SHIFT key and then depressing the note key. Similarly, a note in the lower one-half octave is played by first depressing and releasing the L SHIFT key and then depressing the note key. Two other shift keys are present: UPPER and LOWER. All notes played while the UPPER key is held down will be in the upper octave. Similarly, note F# through B when played while the LOWER key is held down will be in the lower one-half octave. The lower octave notes C through F are not present and depressing any of these 6 keys while the LOWER key is held down or after depressing the L SHIFT key will play the note in the basic octave.

1.B. PLAY STORED TUNE

The 444L-EVAL can play four preprogrammed tunes. Depressing PLAY followed by "⅛", "¼", "½", or "1" will cause one of these tunes to be played. The tunes are:

PLAY 1 —Music Box Dancer

PLAY ½ —Santa Lucia

PLAY ¼ —Godfather Theme

PLAY ⅛ —Theme from Tchaikowsky Piano Concerto #1

1.C. RECORD A TUNE

Any combination of notes and rests up to a total of 48 may be stored in RAM for later replay. A note is stored by depressing the appropriate key(s), followed by the duration of the note (1/16 note, ⅛ note, 3/16 note, ¼ note, ⅜ note, ½ note, ¾ note, whole(1) note), followed by STORE. A rest is stored by selecting the duration and depressing STORE. The rests or durations of 1/16, 3/16, ⅜, and ¾ are obtained by first depressing L SHIFT and then ⅛, ¼, ½, or 1 respectively. When the tune is complete press PLAY followed by STORE. The tune will be played for immediate audition. Subsequent depression of PLAY and then STORE will play the last stored tune.

Only one tune may be stored, regardless of length. Attempts to store a new or second tune will erase the previously stored tune. There are no editing features in this mode. (In a "real system" of this type some form of editing would be desirable. It would not be difficult to add editing features.)

Note: The accuracy of the tones produced is a function of the oscillator accuracy and stability. The crystal oscillator, or an accurate, stable external oscillator is recommended.

2.0. THE 444L-EVAL AS AN UP/DOWN COUNTER/TIMER

By connecting pin L7 to V_{CC} and providing power and oscillator the 444L-EVAL functions as an 8 digit binary/BCD up/down counter. In addition, an approximate 1 Hz signal is produced by the device. The 444L-EVAL can drive a single digit LED display directly. With the appropriate driver (COP472, COP470, MM5450/5451) the device can drive a 4 digit LCD, VF, or LED display. Any combination of these displays can be connected at any given time.

The binary/BCD and and up/down modes are controlled by the states of input pins IN0 and IN2 as indicated below:

IN0 = 1 (Default state) —BCD counter
IN0 = 0 —Binary Counter
IN2 = 1 (Default state) —Count Up
IN2 = 0 —Count Down

The up/down control may be changed at any time. Changing the binary-BCD control during operation clears the counter before counting begins in the new mode.

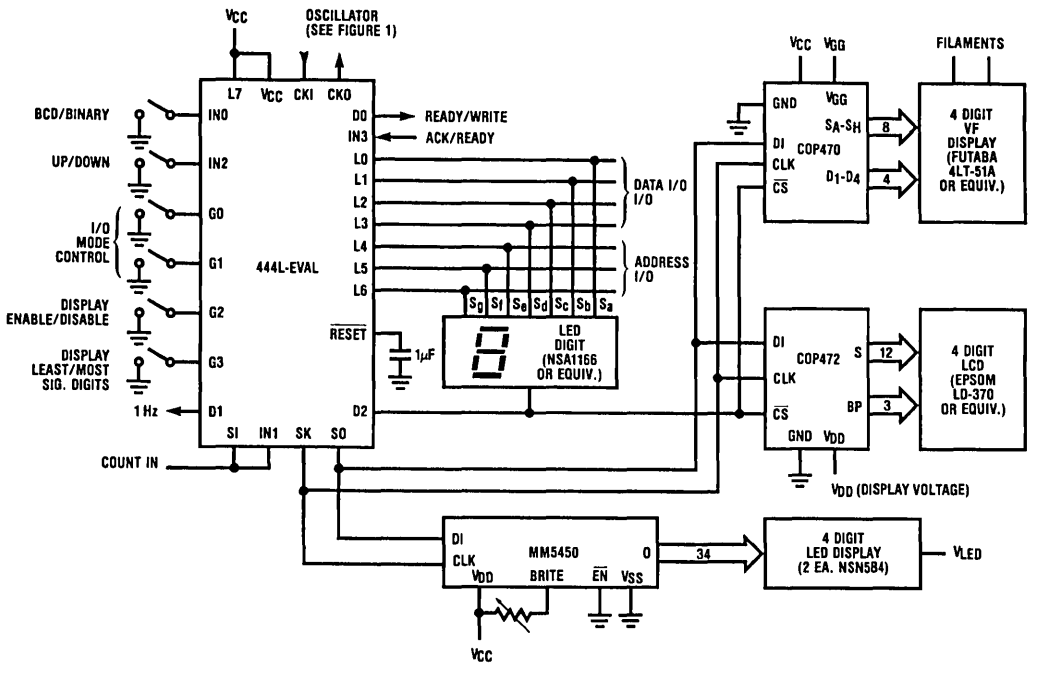
Pins G2 and G3 provide display control to the user. He can choose to view either the most significant 4 digits of the counter or the least significant 4 digits of the counter. Further, the user can disable the update of the 4 digit displays. The controls are as follows:

G2 = 1 (Default state) —Enable update of 4 digit displays
G2 = 0 —Disable update of 4 digit displays
G3 = 1 (Default state) —Display least significant 4 digits of counter
G3 = 0 —Display most significant 4 digits of counter

The single digit LED display displays the least significant digit of the counter. (Note, the direct drive capability for the single digit LED display refers to a small LED digit—NSA1541A, NSA1166k, or equivalent.)

2.A. I/O MODE

The 444L-EVAL has the capability to allow the user to read or write the 8 digit counter through the L port. In the I/O mode, the single digit LED display is disabled. The 4 digit displays are not affected. In this mode pins D0 and IN3 are used for the handshaking sequence. D0 is a Ready/Write signal from the 444L-EVAL to the outside; IN3 is a Write/Acknowledge from the outside to the 444L-EVAL. Data I/O is via L0–L3 with L0 being the least significant bit. Data is standard BCD for the BCD counter mode or standard hex for the binary counter mode. The digit address is on pins L4–L6 with L4 being the least significant bit. Digit address



TL/DD/6937–2

FIGURE 2. 444L-EVAL In Counter Mode

0 is the least significant digit of the counter; digit address 7 is the most significant digit of the counter. The I/O modes are controlled by pins G0 and G1 as follows:

G0	G1	
0	0	Output data with handshake, single digit LED off
0	1	Input data with handshake, single digit LED off
1	0	Auto output, no handshake, single digit LED on
1	1	Default condition, No I/O, single digit LED displays least significant digit of counter

2.A.1. Output Data with Handshake

With this mode selected the 444L-EVAL will output data with a handshake sequence. Note that the outputting of data is relatively slow as the device is counting and updating displays between successive digit outputs.

Before data is output, or the next digit of the counter is output, the 444L-EVAL must see IN3 (Acknowledge or ready from the external world high). The Ready/Write pin (D0) is assumed to be high at this point. With D0 high and IN3 high, the device will output the data and digit address. After the data and address are output, the D0 line—functioning as a write strobe here—goes low. The 444L-EVAL then expects the signal at IN3 to go low indicating that the external world has read the data. When the device sees IN3 go low, D0 will be brought high indicating that the sequence is ready to repeat as soon as IN3 goes high again. The counter digits are output sequentially from least significant digit (digit address 0) through most significant digit (digit address 7). The sequence will continuously repeat as long as this mode is selected.

2.A.2. Input Data with Handshake

The 444L-EVAL will take data supplied to it and load the counter. The sequence is similar to that described above for the output mode. The external device(s) supplies both the data and the digit address where that data is to be loaded.

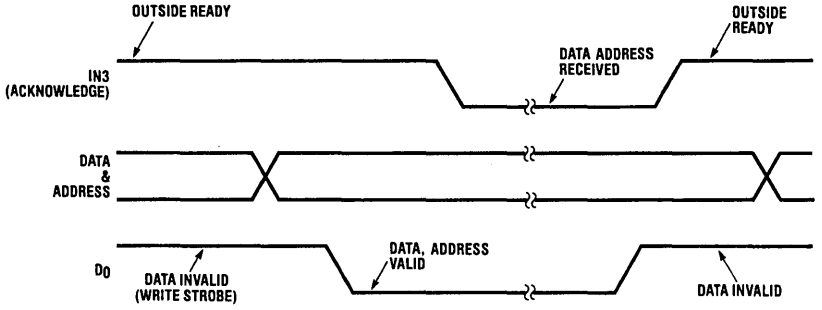
When sending data to the 444L-EVAL, the external circuitry must test that the device is ready to receive data (D0 high). Then the data and address should be presented at the L port. Then the Write signal (IN3) should be driven low. The 444L-EVAL will read the data and then drive D0 low. When D0 goes low, the external circuitry should bring IN3 high. After IN3 returns high, the 444L-EVAL will signal it is ready to receive data by sending D0 high. Note that this sequence is relatively slow. The 444L-EVAL is performing several operations between successive read operations.

2.A.3. Automatic Output Mode

In the automatic output mode, the single digit LED is on. It is not displaying the least significant digit of the counter in this mode. The display is on so that the user can connect this LED digit, select the automatic output mode, and observe the states of the L lines without having to put more sophisticated equipment or circuitry external to the 444L-EVAL. Segments a through d are pins L0 thorugh L3; segments,

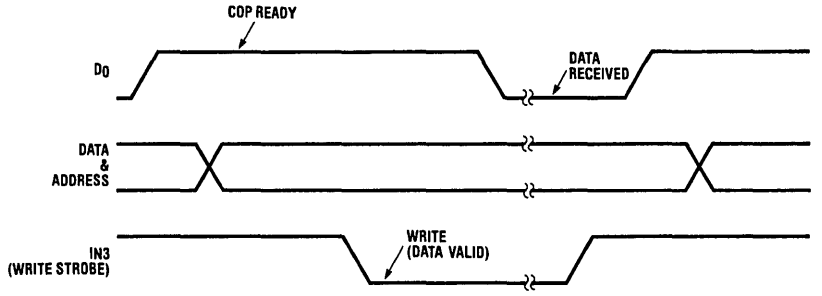
e, f, g are pins L4, L5, and L6. Thus the user can observe the digit address changing and observe the corresponding data.

In this mode, the state of pin IN3 is irrelevant. The 444L-EVAL sequentially outputs the digits of the counter. D0 goes high when the data and address is being changed. D0 goes low when the data is valid. As in the other I/O modes, the process is slow. There is about 4 to 5 milliseconds between the successive digit outputs.



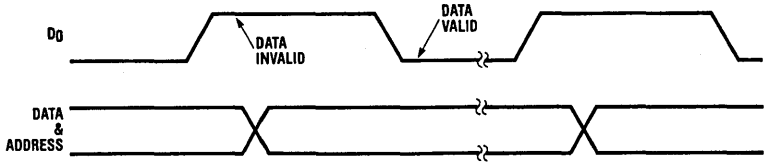
TL/DD/6937-3

FIGURE 3A. Relative Timing—Output Handshake



TL/DD/6937-4

FIGURE 3B. Relative Timing—Input Handshake



TL/DD/6937-5

FIGURE 3C. Relative Timing—Automatic Output

3.0 SELECTED OPTIONS

The 444L-EVAL has the following options selected:

GND	Option 1 = 0	
CKO	Option 2 = 0	CKO is clock generator output to crystal
CKI	Option 3 = 0	CKI oscillator input divide by 32
RESET	Option 4 = 0	Load device to V_{CC} on RESET
L7	Option 5 = 0	Standard output on L7
L6	Option 6 = 2	High current LED direct segment drive on L6
L5	Option 7 = 2	High current LED direct segment drive on L5
L4	Option 8 = 2	High current LED direct segment drive on L4
IN1	Option 9 = 0	Load device to V_{CC} on IN1
IN2	Option 10 = 0	Load device to V_{CC} on IN2
V_{CC}	Option 11 = 1	4.5V to 9.5V operation
L3	Option 12 = 2	High current LED direct segment drive on L3
L2	Option 13 = 2	High current LED direct segment drive on L2
L1	Option 14 = 2	High current LED direct segment drive on L1
L0	Option 15 = 2	High current LED direct segment drive on L0
SI	Option 16 = 0	Load device to V_{CC} on SI
SO	Option 17 = 2	Push-pull output on SO
SK	Option 18 = 2	Push-pull output on SK
IN0	Option 19 = 0	Load device to V_{CC} on IN0
IN3	Option 20 = 0	Load device to V_{CC} on IN3
G0	Option 21 = 0	Very high current standard output on G0
G1	Option 22 = 2	High current standard output on G1
G2	Option 23 = 4	Standard LSTTL output on G2
G3	Option 24 = 4	Standard LSTTL output on G3
D3	Option 25 = 0	Very high current standard output on D3
D2	Option 26 = 0	Very high current standard output on D2
D1	Option 27 = 0	Very high current standard output on D1
D0	Option 28 = 0	Very high current standard output on D0
	Option 29 = 0	Standard TTL input levels on L
	Option 30 = 0	Standard TTL input levels on IN
	Option 31 = 0	Standard TTL input levels on G
	Option 32 = 0	Standard TTL input levels on SI
	Option 33 = 1	Schmitt trigger inputs on RESET
	Option 34 = 0	CKO input levels, not used here
	Option 35 = 0	COP444L
	Option 36 = 0	Normal RESET operation

4.0 CONCLUSION

The 444L-EVAL demonstrates much of the capability of the COP444L. It does not indicate the limits of the device by any means. The I/O features were included to demonstrate that capability. The fact that they are slow is due strictly to the program. If such I/O capability were a necessary part of an application it could be accomplished much much faster than was done here. The counter modes are quite versatile and are generally self explanatory. It was fairly easy to provide a counter with the versatility of that included here. The music synthesis mode demonstrates clearly the program efficiency of the device.

The 444L-EVAL is intended for demonstration. There is no question that aspects of its operation could be improved and tailored to a specific application. It is unlikely that this particular combination of features would be found in any one application. It is also interesting to note that the program memory in the device is not full. There is still a significant amount of room left in the ROM. This should serve to make it clear that the capabilities of the device have not been stretched at all in order to include these demonstration functions.

Oscillator Characteristics of COPS™ Microcontrollers

National Semiconductor
COP Note 5

Table of Contents

1.0 INTRODUCTION

COPS microcontrollers will operate with a wide variety of oscillator circuits. This paper focuses on two of the oscillator options available on COPS microcontrollers: the internal RC oscillator, and the crystal or inverter oscillator. The typical behavior of the RC oscillator with temperature and voltage (and typical values of R and C) is documented. For the crystal or inverter option, circuit configurations (RC, RL, RLC, R, LC, L) are presented which will allow the microcontroller to operate properly without the use of ceramic resonator or crystal.

The passive components used were inexpensive, uncompensated devices: standard carbon resistors, ceramic or foil capacitors, and air core or iron core inductors. To provide reasonably clear data on the characteristics of the microcontroller itself, no attempt at compensation for the external components was made.

2.0 RC OSCILLATOR OPTION

With the RC oscillator option selected, the graphs in *Figures 1* through *6* indicate the variation of the instruction cycle time of the microcontroller with temperature and voltage. Typical R and C values, as recommended in the respective device data sheets, were used. The graphs are composite graphs reflecting the worst case variations of the devices tested. Therefore, the graphs show a percentage change of the instruction cycle time from a base or reference value. Where the results are plotted against voltage the reference is the value at V_{CC} = 5V. Where the results are plotted against temperature, the reference is the value at T = 20°C. A positive percent variation indicates a longer instruction cycle time and therefore a slower oscillator frequency. Similarly, a negative percent variation indicates a shorter instruction cycle time and therefore a faster oscillator frequency.

The measurements were taken by holding the RESET pin of the device low and measuring the period of the waveform at pin SK. In this mode the SK period is the instruction cycle time. For divide by 4 the oscillator frequency is given by the following:

$$\text{frequency} = \frac{4}{\text{SK period}}$$

Measurements were taken at temperatures between −40°C and +85°C and at V_{CC} values between 4.5V and 9.5V. However, the reader must remember that the COP400 series is specified only between 0°C and +70°C. The reader must also remember that the COP420 is specified at V_{CC} levels between 4.5V and 6.3V only. The data here is usable for the COP300 series, which is specified at the extended temperature range of −40°C to +85°C. However, the reader must keep in mind the generally more restricted V_{CC} range for some of the various COP300 series microcontrollers.

The graphs in *Figures 1* through *6* reflect the variation of the microcontroller only. The resistor and capacitor were not in the temperature chamber with the COPS device. Obviously, the results will be affected by the variation of the R and C with temperature. However, this can vary dramatically with the type of components used. The user will have to combine the data here with the characteristics of the external components used to determine what type of variation may be expected in his system.

3.0 CRYSTAL OR INVERTER OPTION

With the crystal or inverter option selected on the COPS microcontroller there is, effectively, an inverter between the CKI and CKO pins. CKI is the input to the inverter and CKO is the output. Various passive circuits were connected between CKI and CKO and the results documented. Of the operational circuits, a subset was tested over temperature with the microcontroller only in the temperature chamber. A smaller subset was tested over temperature with both the microcontroller and the oscillator network in the temperature chamber.

The data with the oscillator network in the temperature chamber is obviously highly dependent on the particular components used. This data was taken with standard, inexpensive, uncompensated components. Neither high precision nor high stability components were used. This data is included only to provide the user with some very general indication of how the oscillator frequency may vary with temperature in a real system.

3.1 COP420/COP402

Except for the ROM, the COP420 and COP402 are equivalent devices. The internal circuitry of each device is identical. Therefore, data taken for one of the devices is equally

applicable to the other. The following discussion will refer to the COP420 but all such references apply equally well to the COP402. Similarly, the graphs for the COP420 apply to the COP402 and *vice versa*.

With the crystal option selected, the COP420 oscillator circuitry will readily oscillate with almost any circuit configuration between CKI and CKO. What difficulty there is lies in finding the network of the device. With the appropriate divide option selected, oscillator frequencies between 800 kHz and 4 MHz are valid for the COP420. No data was taken for any network that produced an oscillation frequency outside the valid range.

3.1.1 L, LC, and RLC Networks

Various L, LC, and RLC networks were connected with varying results. Certain networks produced results much more stable than the RC networks; others were no better than the RC networks. With a single inductor connected between CKI and CKO, frequencies between 1 MHz and 4 MHz were easily obtained. However, the input gate capacitance at CKI (typically 5 pF to 10 pF) and the series resistance of the inductance become factors that impact the oscillation frequency and its stability over temperature.

The addition of a capacitor between CKI and ground tends to reduce the effects of the internal gate capacitance. For the single L, single C network of this type, the capacitor value should be greater than about 50 pF to begin to effectively swamp out the effects of the input gate capacitance. As might be expected, LC combinations which had their resonant frequencies within the valid COP420 frequency range produced the best results.

The addition of another capacitor(s) to the basic two-component LC network, as shown in *Figure III.1*, produced very good results. Varying the capacitor values in these networks — especially those capacitors between CKI and ground and CKO and ground — provided a great deal of control over the oscillation frequency. In *Figure III.1*, varying C1 from 25 pF to 0.01 μF produced oscillation frequencies between about 3 MHz and 1.6 MHz (C2 = 25 pF, L = 56 μH). In *Figure III.2*, with C1 = 330 pF, L = 56 μH, and C2 = 27 pF, varying C3 between 10 pF and 0.003 μF produced oscillation frequencies between about 2 MHz and 1.1 MHz. Varying C2 in *Figure 111.3* produced a similar kind of control.

As the graphs indicate, various types of RLC networks were also tried. The range of possible usable circuits here is limited only by the user's imagination and his favorite type of RLC oscillator circuit. When their resonant frequency is within the valid frequency range of the COP420, LC and RLC networks can be a very effective substitute for a crystal. The only potential problem is that a good RLC, or even LC, oscillator circuit may not be a cost-effective substitute for a crystal in a COP420 system. The user will have to make that determination.

3.2 COP420L

The valid input frequency range for the COP420L, with the appropriate divide option selected, is between 200 kHz and 2.097 MHz. With the crystal option selected the COP420L oscillated much less readily than the COP420.

The LC networks gave outstanding results with the COP420L. With the simple two-component LC network shown in the graphs, holding C at 50 pF and varying L from 200 μH to 700 μH gave oscillation frequencies from about 2 MHz to 1 MHz. Holding L at 390 μH and varying C from 10 pF to 700 pF gave oscillation frequencies of about 2 MHz to 1.6 MHz. Similar results were obtained when a capacitor was placed in parallel with the inductance.

3.3 COP410L

The COP410L has a valid input frequency range of 200 kHz to 530 kHz.

The LC networks also gave very good results. With the simple LC network shown in the graphs, holding L at 4700 μH and varying C from 25 pF to 0.003 μF gave oscillation frequencies of about 460 kHz to 225 kHz.

3.4 GENERAL NOTES

With the crystal or inverter option selected on COPS microcontrollers, a wide variety of networks may be used in place of the ceramic resonator or crystal.

LC and RLC networks can be used in any of the devices. Appropriately designed, these networks will provide a stable oscillation frequency for the microcontroller. The user will have to allow for the variation of the external components with temperature when using these networks. The problems with networks such as these is that they may not be cost-effective alternatives to the crystal or resonator, especially if high stability, temperature compensated components are used. The user will have to make the determination of cost-effectiveness.

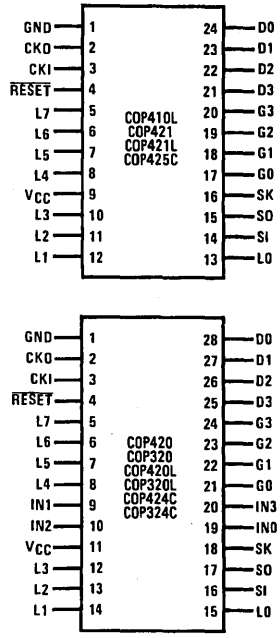
A final note is that all of these networks place a load on the CKO output. If the signal from CKO is needed elsewhere in the system and a circuit similar to one of those discussed in this document is used, it will probably be necessary to buffer the CKO output.

4.0 Conclusion

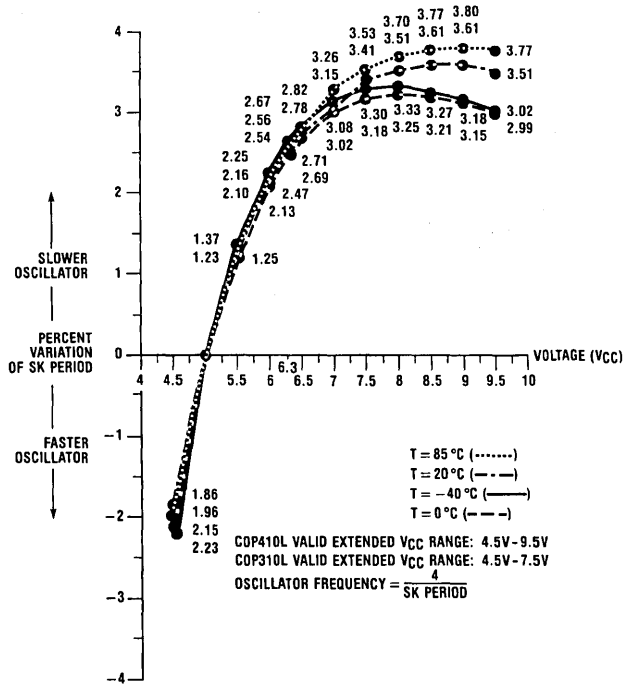
The networks described are generally simple and inexpensive and have all been observed to be functional.

The results obtained provide greater flexibility in the oscillator selection in a COPs system and gives the user some general indication as to what may be expected with the various circuits described.

COP Microcontroller Pinouts



TL/DD/6938-1



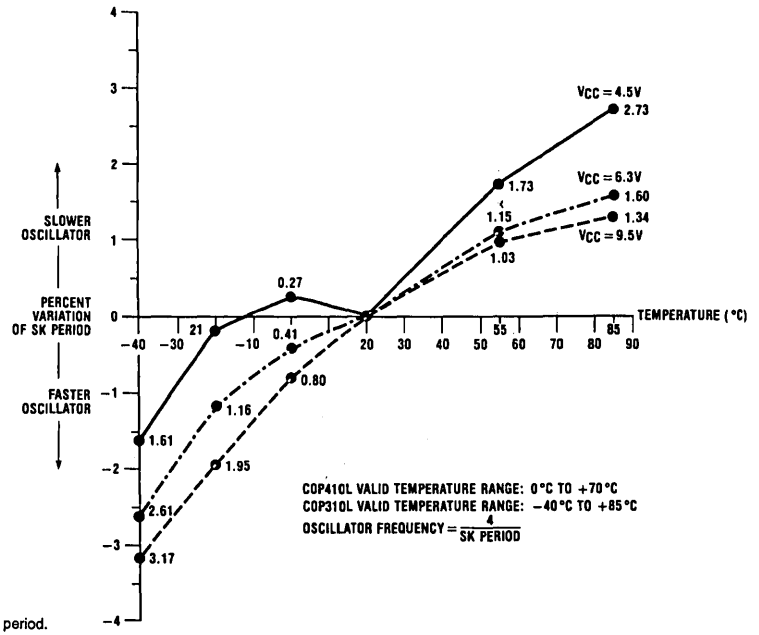
TL/DD/6938-2

Note 1: Base period at V_{CC} = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 1. COP310L/COP410L RC Oscillator Variation with V_{CC}



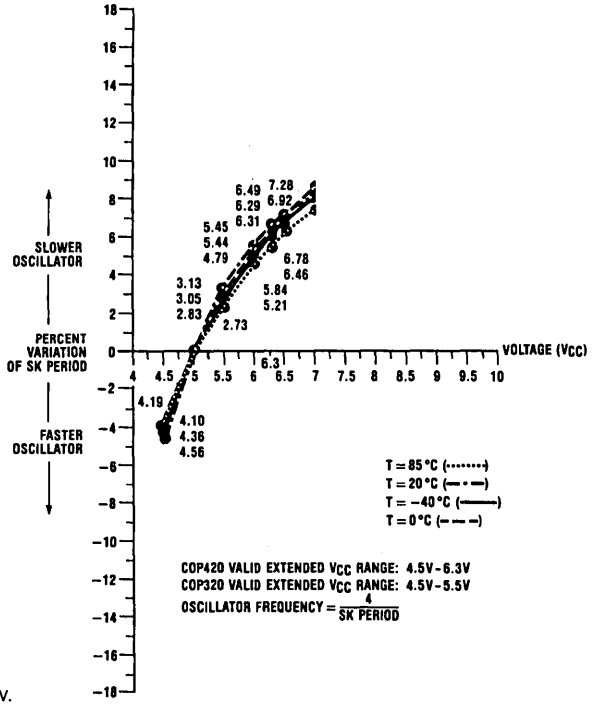
Note 1: 20°C = base period.

TL/DD/6938-3

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 2. COP310L/COP410L RC Oscillator Variation with Temperature



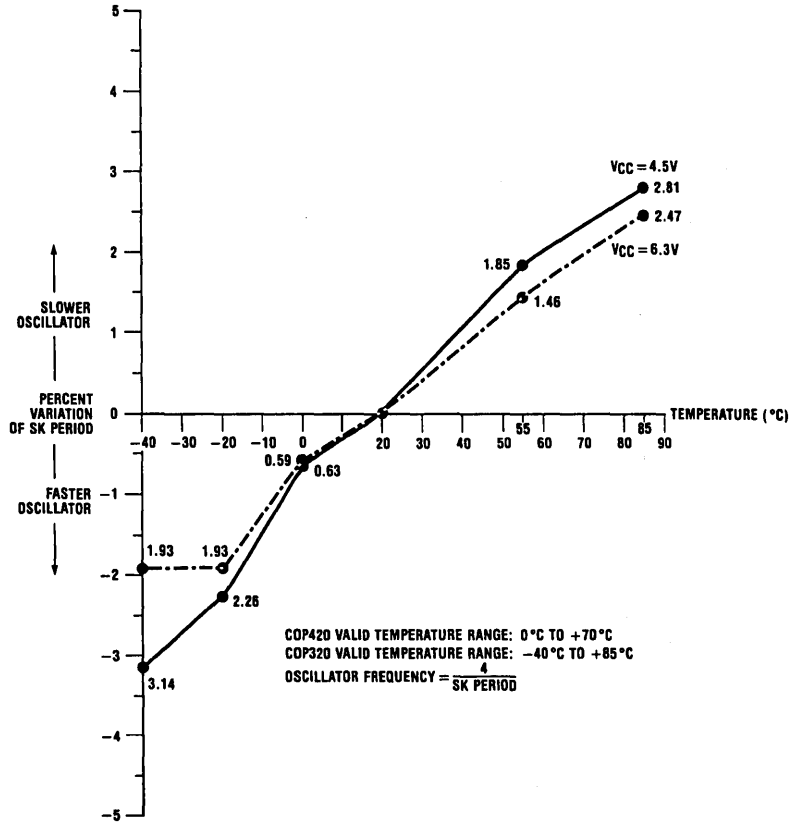
Note 1: Base period at V_{CC} = 5.0V.

TL/DD/6938-4

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 3. COP320/COP420 RC Oscillator Variation with V_{CC}



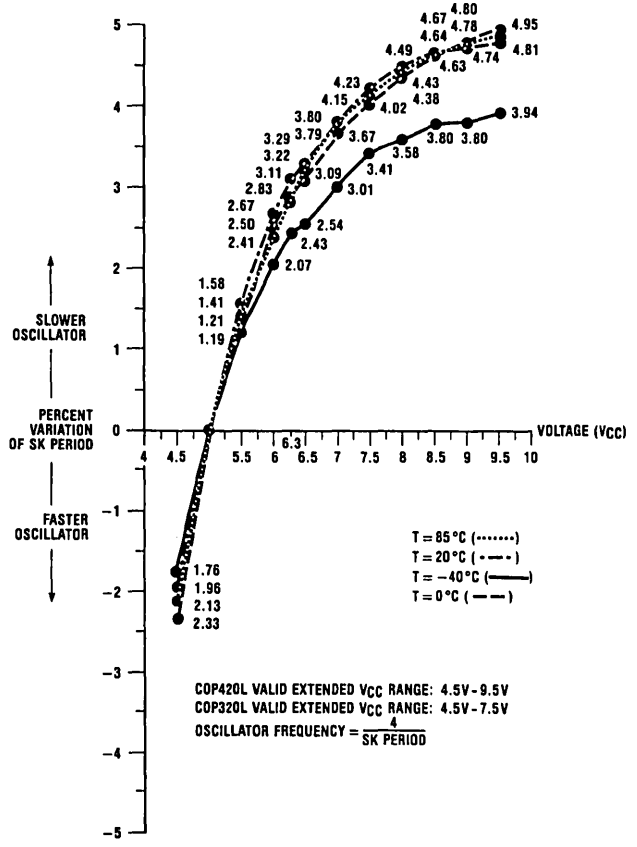
TL/DD/6938–5

Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 4. COP320/COP420 RC Oscillator Variation with Temperature

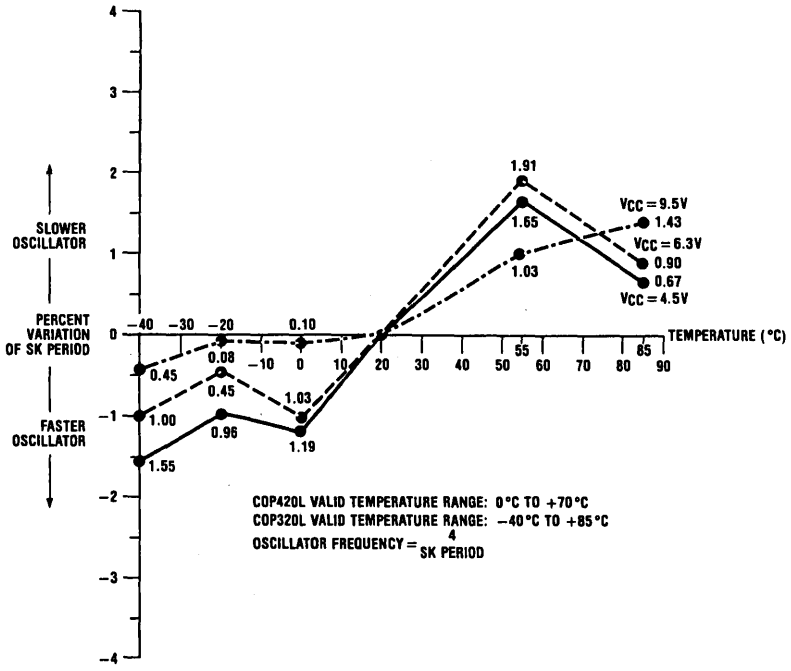


Note 1: Base period at V_{CC} = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 5. COP320L/COP420L RC Oscillator Variation with V_{CC}



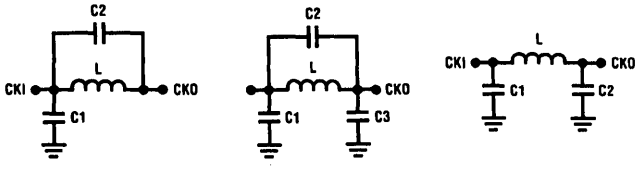
TL/DD/6938-7

Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

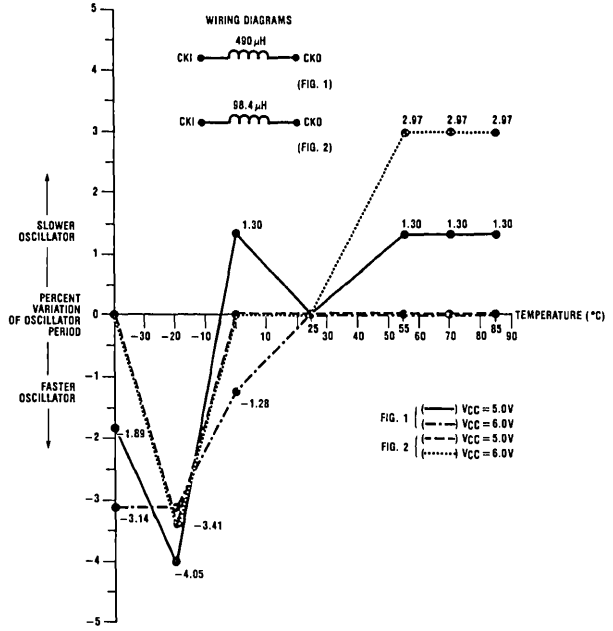
FIGURE 6. COP320L/COP420L RC Oscillator Variation with Temperature



TL/DD/6938-8

FIGURE III.1 **FIGURE III.2** **FIGURE III.3**

COP402



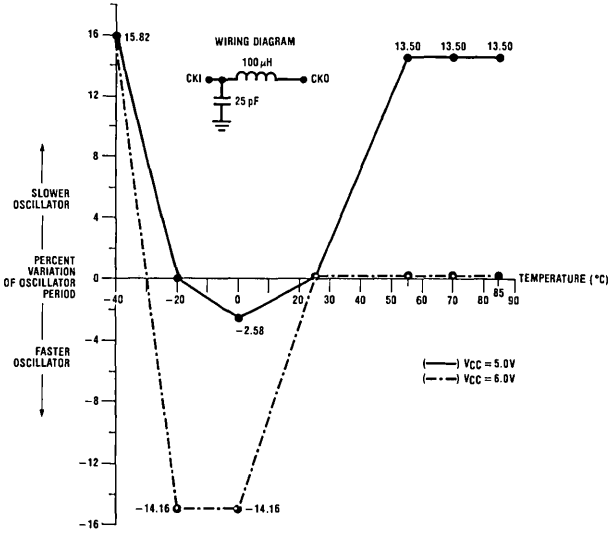
Note 1: 25°C = base period.

TL/DD/6938–9

Note 2: Device variation only. Graph does not include "L" variation with temperature.

FIGURE 7

COP402



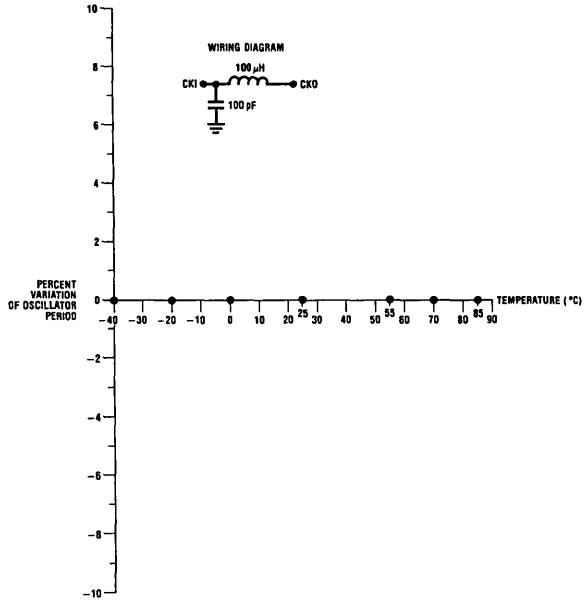
Note 1: 25°C = base period.

TL/DD/6938–10

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 8

COP420



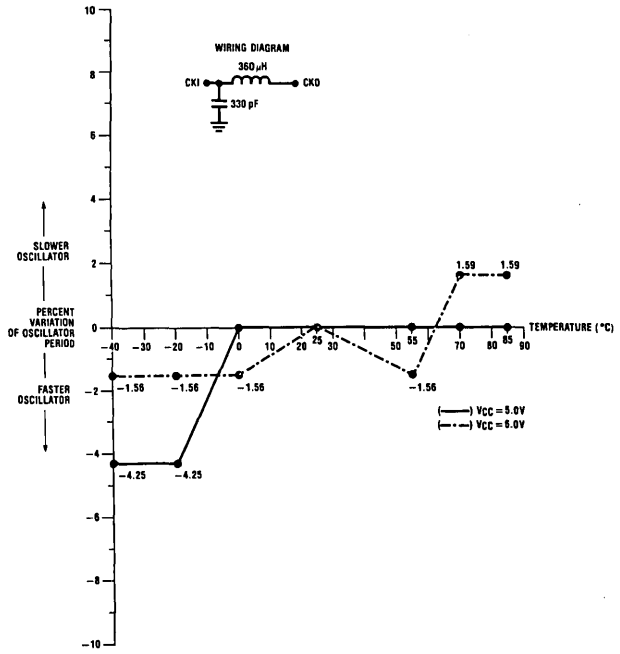
TL/DD/6938–11

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.
No measurable variation over temperature.

FIGURE 9

COP420



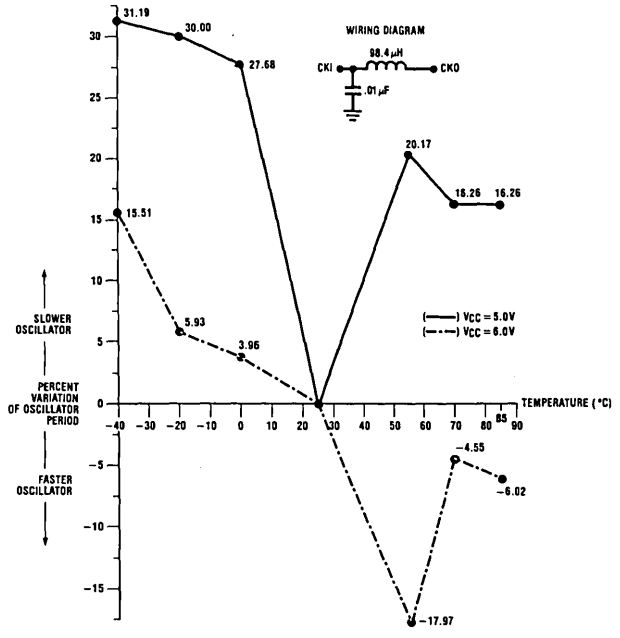
TL/DD/6938–12

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 10

COP402



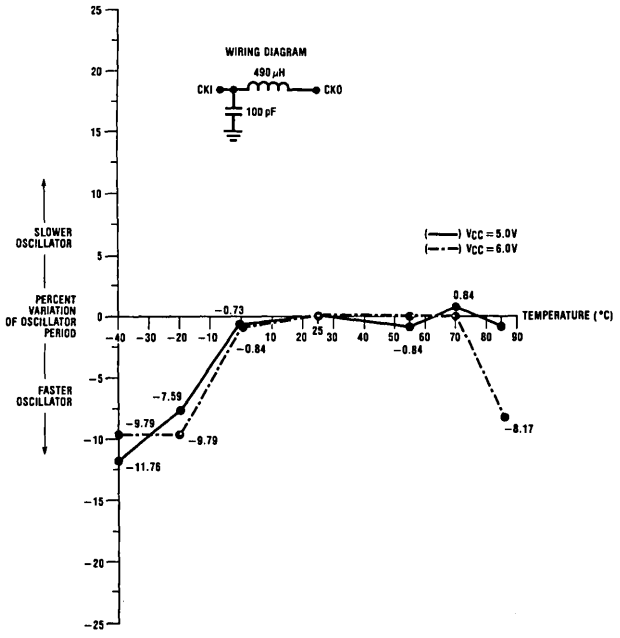
TL/DD/6938-13

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 11

COP402



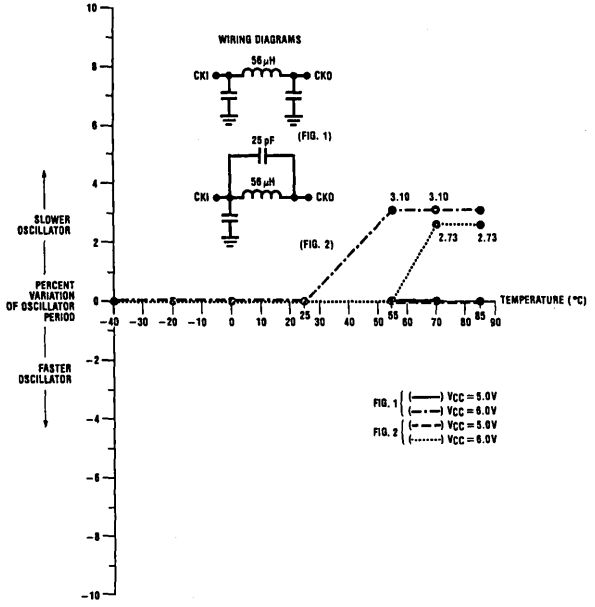
TL/DD/6938-14

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 12

COP402



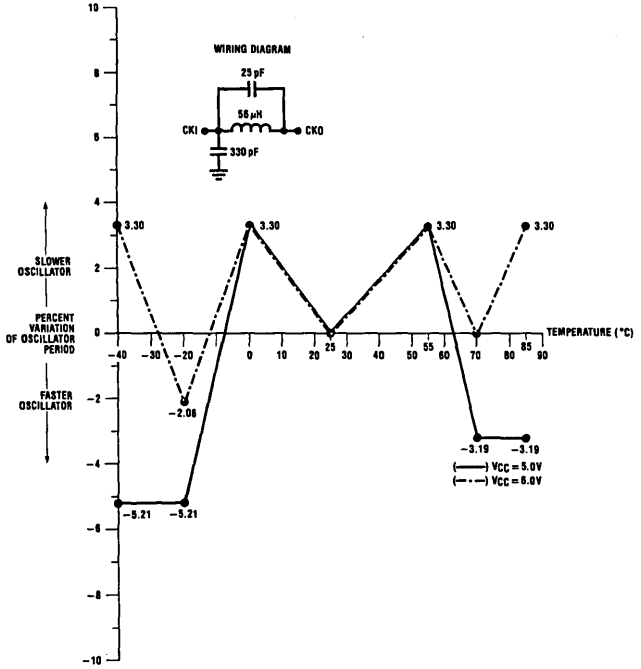
TL/DD/6938–15

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 13

COP402



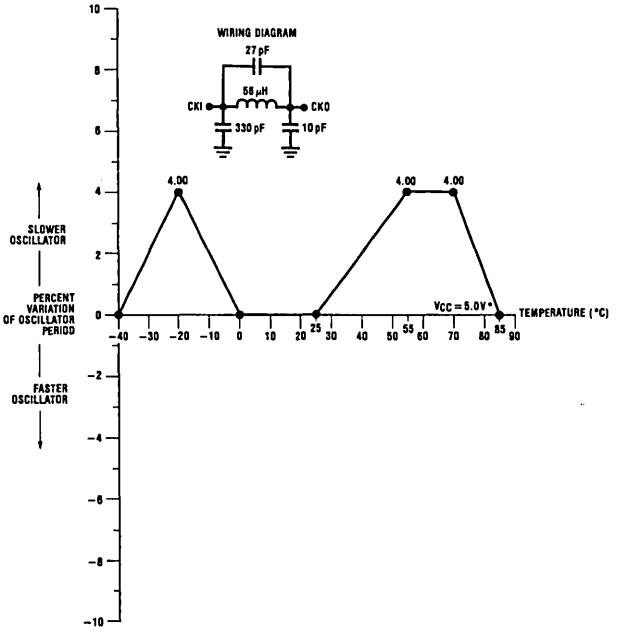
TL/DD/6938–16

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 14

COP402



TL/DD/6938-17

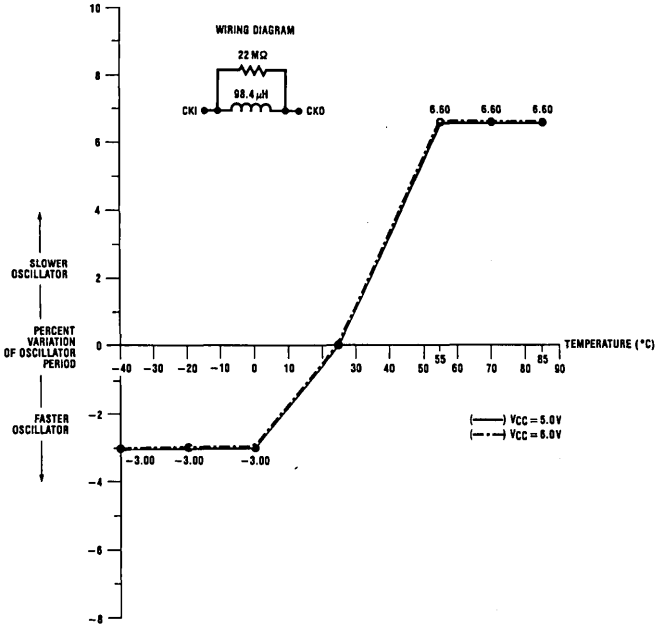
Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

*No variation at 6V.

FIGURE 15

COP402



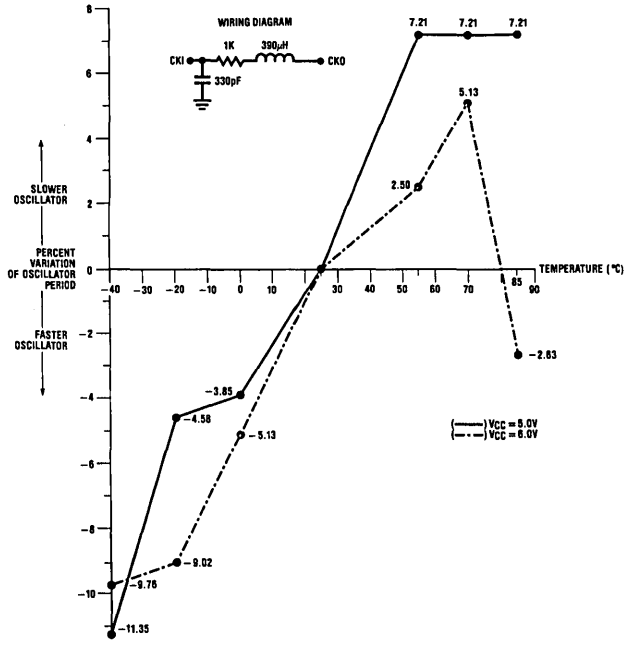
TL/DD/6938-18

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RL variation with temperature.

FIGURE 16

COP402



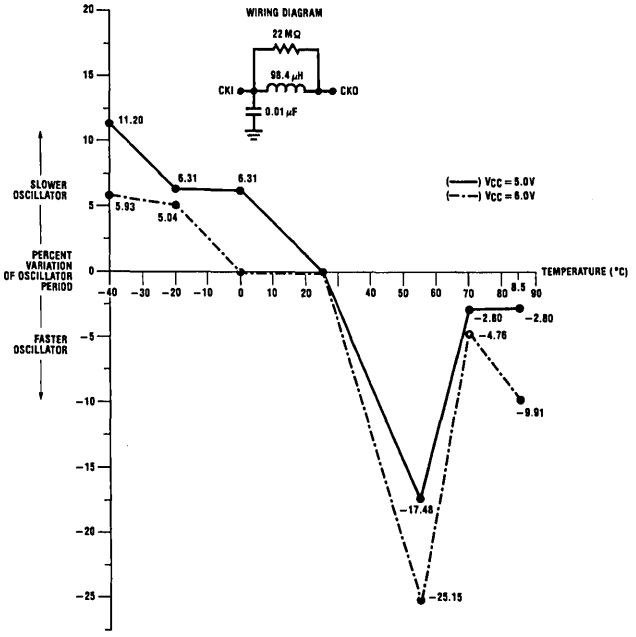
TL/DD/6938–19

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 17

COP402



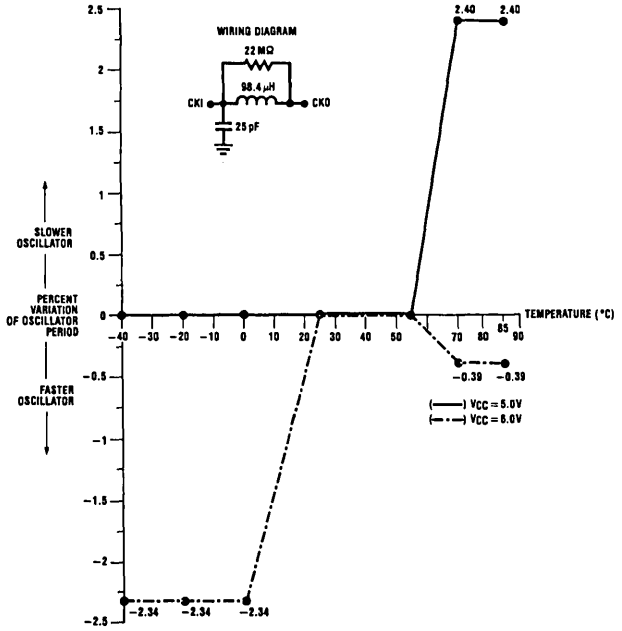
TL/DD/6938–20

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 18

COP402

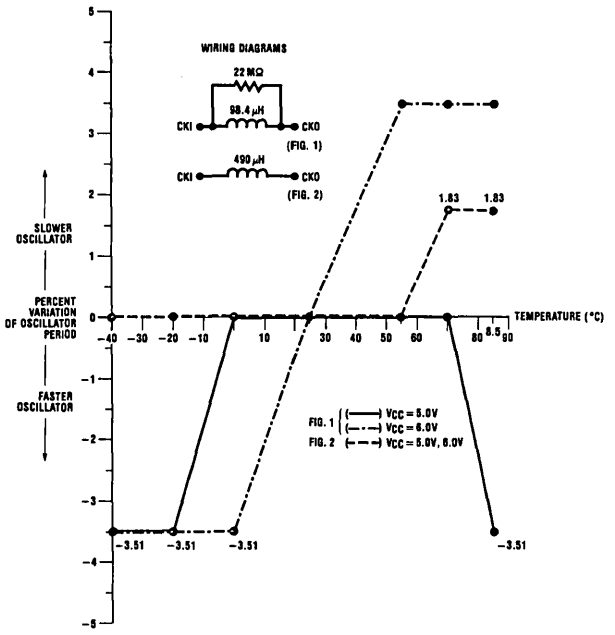


Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 19

COP402

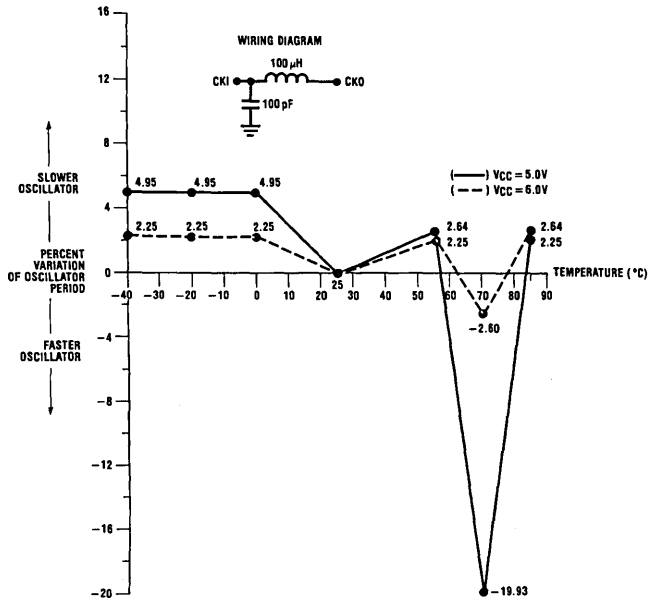


Note 1: 25°C = base period.

Note 2: RL in oven with COP402.

FIGURE 20

COP420



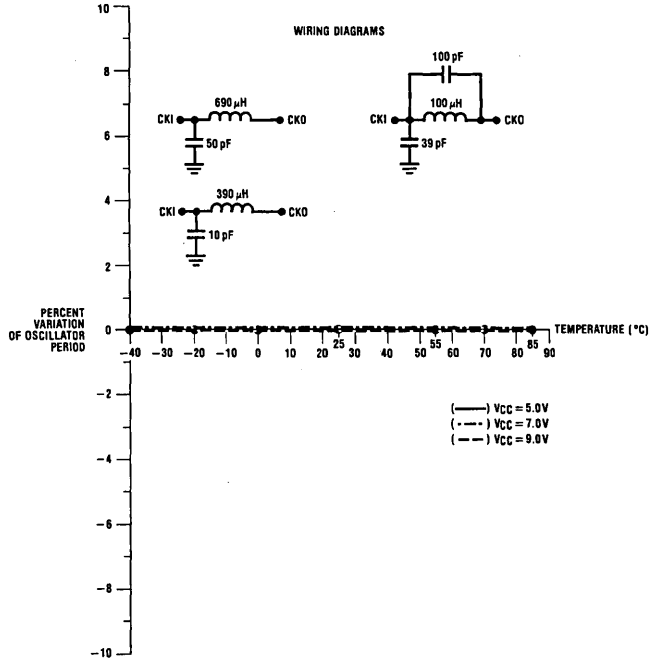
TL/DD/6938-23

Note 1: 25°C = base period.

Note 2: LC in oven with COP402.

FIGURE 21

COP420L



TL/DD/6938-26

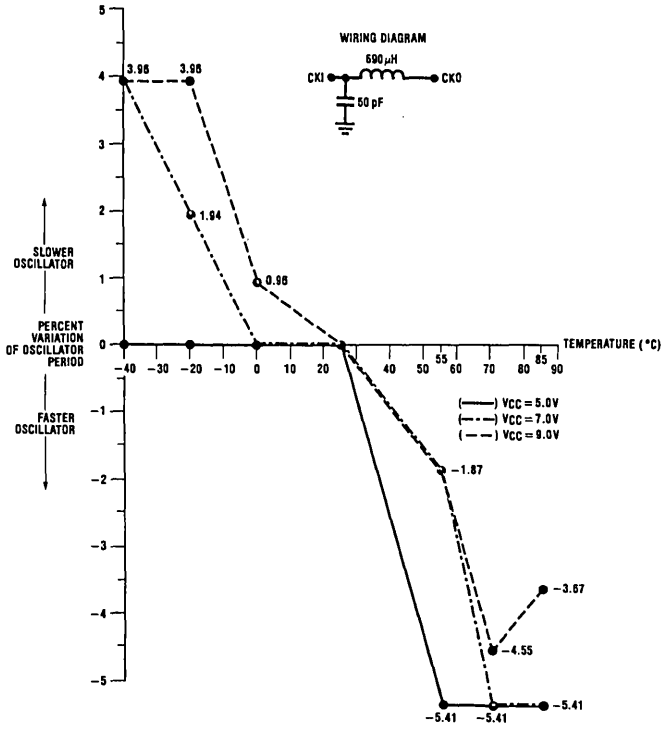
Note 1: No measurable variation for all three circuits above.

Note 2: 25°C = base period.

Note 3: Device variation only. Graph does not include LC variation with temperature.

FIGURE 22

COP420L



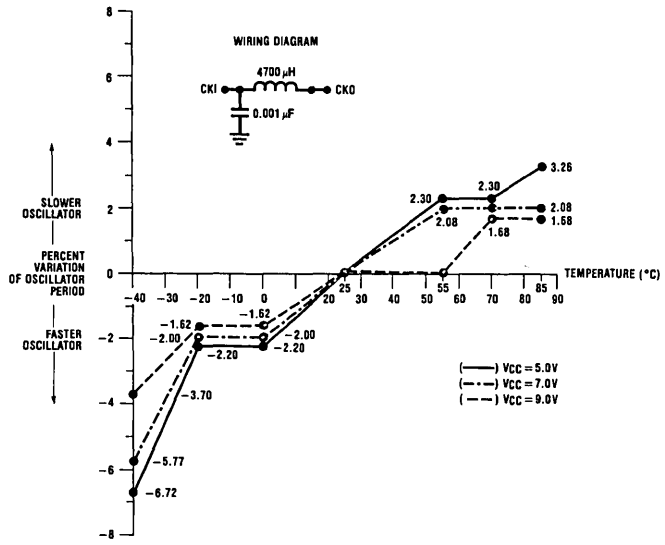
TL/DD/6938-28

Note 1: 25°C = base period.

Note 2: LC in oven with COP420L.

FIGURE 23

COP410L



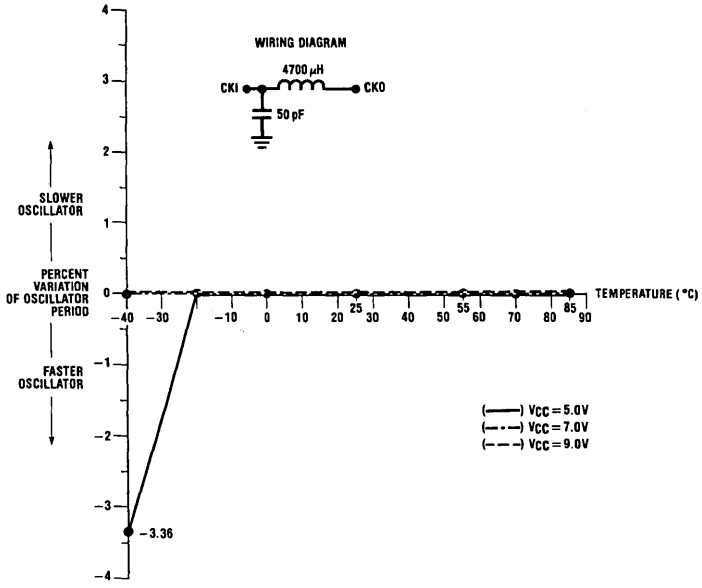
TL/DD/6938-29

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 24

COP410L



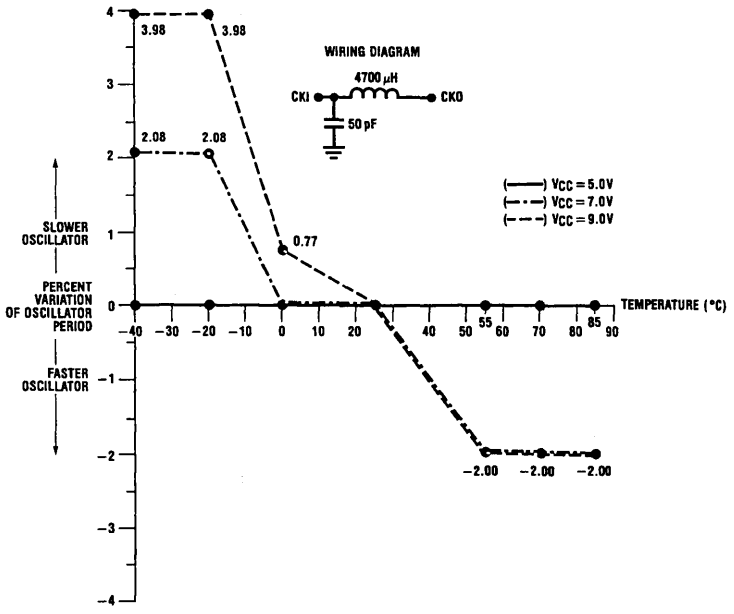
TL/DD/6938–30

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 25

COP410L



TL/DD/6938–32

Note 1: 25°C = base period.

Note 2: LC in oven with COP410L.

FIGURE 26

Triac Control Using the COP400 Microcontroller Family

National Semiconductor
COP Note 6

Table of Contents

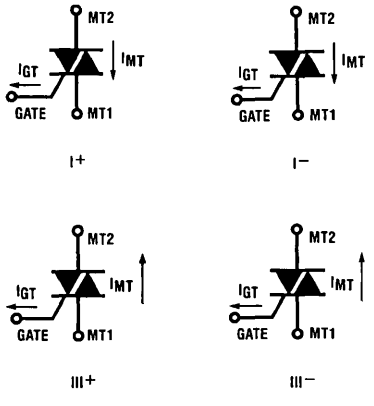
1.0 Triac Control

The COP400 single-chip controller family members provide computational ability and speed which is more than adequate to intelligently manage power control. These controllers provide digital control while low cost and short turnaround enhance COPS™ desirability. The COPS controllers are capable of 4 μs cycle times which can provide more than adequate computational ability when controlling 60 Hz line voltage. Input and output options available on the COPS devices can contour the device to apply in many electrical situations. A more detailed description of COPS qualifications is available in the COP400 data sheets.

The COPS controller family may be utilized to manage power in many ways. This paper is devoted to the investigation of low cost triac interfaces with the COP400 family microcontroller and software techniques for power control applications.

1.1 BASIC TRIAC OPERATION

A triac is basically a bidirectional switch which can be used to control AC power. In the high-impedance state, the triac blocks the principal voltage across the main terminals. By pulsing the gate or applying a steady state gate signal, the triac may be triggered into a low impedance state where conduction across the main terminals will occur. The gate signal polarity need not follow the main terminal polarity; however, this does affect the gate current requirements. Gate current requirements vary depending on the direction of the main terminal current and the gate current. The four trigger modes are illustrated in *Figure 1.*

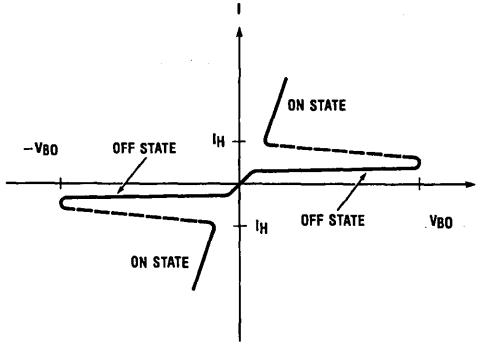


TL/DD/6939–1

FIGURE 1. Gate Trigger Modes. Polarities Referenced to Main Terminal 1.

The breakover voltage (V_{BO}) is specified with the gate current (I_{GT}) equal to zero. By increasing the gate current supplied to the triac, V_{BO} can be reduced to cause the triac to go into the conduction or on state. Once the triac has entered the on state the gate signal need not be present to sustain conduction. The triac will turn itself off when the main terminal current falls below the minimum holding current required to sustain conduction (I_H).

A typical current and voltage characteristic curve is given in *Figure 2*. As can be seen, when the gate voltage and the main terminal 2 (MT2) voltages are positive with respect to MT1 the triac will operate in quandrant 1. In this case the trigger circuit sources current to the triac (I+ MODE).



TL/DD/6939–2

FIGURE 2. Voltage-Current Characteristics

After conduction occurs the main terminal current is independent of the gate current; however, due to the structure of the triac the gate trigger current is dependent on the direction of the main terminal current. The gate current requirements vary from mode to mode. In general, a triac is more easily triggered when the gate current is in the same direction as the main terminal current. This can be illustrated in the situation where there is not sufficient gate drive to cause conduction when MT2 is both positive and negative. In this case the triac may act as a single direction SCR and conduction occurs in only one direction. The trigger circuit must be designed to provide trigger currents for the worst case trigger situation. Another reason ample trigger current must be supplied is to prevent localized heating within the pellet and speed up turn-on time. If the triac is barely triggered only a small portion of the junction will begin to conduct, thus causing localized heating and slower turn-on. If an insufficient gate pulse is applied damage to the triac may result.

1.2 TRIGGERING

Gate triggering signals should exceed the minimum rated trigger requirements as specified by the manufacturer. This is essential to guarantee rapid turn-on time and consistent operation from device to device.

Triac turn-on time is primarily dependent on the magnitude of the applied gate signal. To obtain decreased turn-on times a sufficiently large gate signal should be applied. Faster turn-on time eliminates localized heat spots within the pellet structure and increases triac dependability.

Digital logic circuits, without large buffers, may not have the drive capabilities to efficiently turn on a triac. To insure proper operation in all firing situations, external trigger circuitry might become necessary. Also, to prevent noise from disturbing the logic levels, AC/DC isolation or coupling techniques must be utilized. Sensitive gate triacs which require minimal gate input signal and provide a limited amount of main terminal current may be driven directly. This paper will focus on $120V_{AC}$ applications of power control.

1.3 ZERO VOLTAGE DETECTION

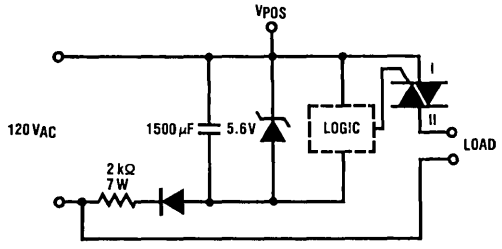
In many applications it is advantageous to switch power at the AC line zero voltage crossing. In doing this, the device being controlled is not subjected to inherent AC transients. By utilizing this technique, greater dependability can be obtained from the switching device and the device being switched. It is also sometimes desirable to reference an event on a cyclic basis corresponding to the AC line frequency. Depending on the load characteristics, switching times need to be chosen carefully to insure optimal performance. Triac controlled AC switching referenced to the AC 60 Hz line frequency enables precise control over the conduction angle at which the triac is fired. This enables the COPS device to control the power output by increasing or decreasing the conduction angle in each half cycle.

A wide variety of zero voltage detection circuits are available in various levels of sophistication. COPS devices, in most cases, can compensate for noisy or semi-accurate ZVD circuits. This compensation is utilized in the form of debounce and delay routines. If a noisy transition occurs near zero volts the COPS device can wait for a valid transition period specified by the maximum amount of noise present. Some software considerations are presented in the software section and are commented upon. The minimal detection circuit is shown in *Figure 9*.

1.4 DIRECT COUPLE

Isolation associated problems can be overcome by means of direct AC coupling. One such method is illustrated in *Figure 3*. This circuit incorporates a half-wave rectifier in conjunction with a filter capacitor to provide the logic power supply. The positive half-cycle is allowed to drop across the zener diode and be filtered by the capacitor. This creates a low cost line interface; however, only a limited supply current is available. In order to control the current capabilities of this circuit the series resistor must be modified. However, as more current is required, the power that must be dissipated in the series resistor increases. This increases the power dissipation requirements of the series resistor and the system cost. For applications which require large current sources an alternative method is advisable. In order to assure consistent operation, power supply ripple must be mini-

mized. COPS devices can be operated over a relatively wide power supply range. However, excessive ripple may cause an inadvertent reset operation of the device.

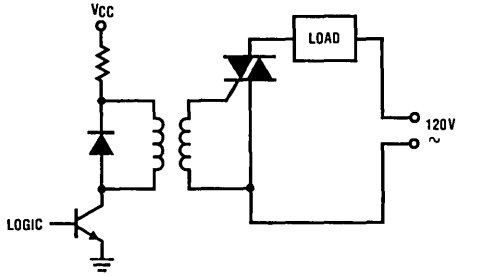


TL/DD/6939–3

FIGURE 3. AC Direct Couple

1.5 PULSE TRANSFORMER INTERFACE

Digital logic control of triacs is easily accomplished by triggering through pulse transformers or optical coupling. The energy step-up gained by using a pulse transformer should provide a more than adequate gate trigger signal. This complies with manufacturers' suggested gate signal requirements. Pulse transformers also provide AC/DC isolation necessary in control logic interfaces. Minimal circuit interface to the pulse transformer is required as shown in *Figure 4.* Optical coupling circuits provide isolation, and in some cases adequate gate drive capabilities.



TL/DD/6939–4

FIGURE 4. Pulse Transformer Interface

A logic controlled pulse is applied to the base of the transistor to switch current through the primary of the pulse transformer. The transformer then transfers the signal to the secondary and causes the triac to fire. The energy transfer that is now available on the secondary is more than adequate to turn on the triac in any of its operating modes. When the pulse transformer is switched off a reverse EMF is generated in the primary coil which may cause damage to the transistor. The diode across the primary serves to protect the collector junction of the switching transistor. Another major advantage is AC isolation; the gate of the triac is now completely isolated from the logic portion of the circuit.

1.6 FALSE TURN-ON

When switching an inductive load, voltage spikes may be generated across the main terminals of the triac which have the potential of a non-gated turn-on of the triac. This creates the undesirable situation of limited control of the system. In a system with an inductive load the voltage leads the current by a phase shift corresponding to the amount of inductance in the motor. As the current passes near zero, the voltage is at a non-zero value, offset due to the phase shift. When the principal current through the triac pellet decreases to a value not capable of sustaining conduction the triac will turn off. At this point in time the voltage across the terminals will instantaneously attain a value corresponding to the phase shift caused by the inductive load. The rapid decay of current in the inductor causes an L dI/dT voltage applied across the terminals of the triac. Should this voltage exceed the blocking voltage specified for the triac, a false turn-on will occur.

In order to avoid false turn-on, a snubber network must be added across the terminals to absorb the excess energy generated by this situation. A common form of this network is a simple RC in series across the terminals. In order to select the values of the network it is necessary to determine the peak voltage allowable in the system and the maximum dV/dT stress the triac can withstand. One approach to obtaining the optimal values for R_S and C_S is to model the effective circuit and solve for the triac voltage. The snubber in conjunction with the load can now be modeled as an RLC network. Due to the two storage elements (L motor, C snubber) a second order differential equation is generated. Rather than approach this problem from a computer standpoint it becomes much easier to obtain design curves generated for rapid solution of this problem. These design curves are available in many triac publications. (For instance, see RCA application note AN 4745.)

2.0 Software Techniques

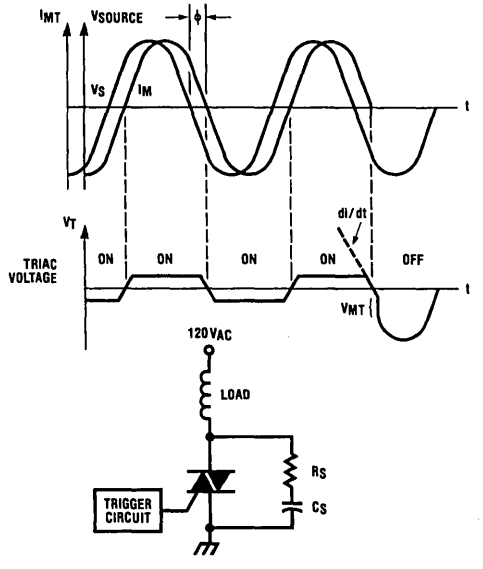
2.1 ZERO VOLTAGE DETECTION

In order to intelligently control triacs on a cyclic basis, an accurate time base must be defined. This may be in the form of an AC, 60 Hz sync pulse generated by a zero voltage detection circuit or a simple real time clock. The COP400 series microcontrollers are suited to accommodate either of these time base schemes while accomplishing auxiliary tasks.

Zero voltage detection is the most useful scheme in AC power control because it affords a real time clock base as well as a reference point in the AC waveform. With this information it is possible to minimize RFI by initiating power-on operations near the AC line voltage zero crossing. It is also possible to fire the triac for only a portion of the cycle, thus utilizing conduction angle manipulation. This is useful in both motor control and light intensity control.

Sophisticated zero voltage detection circuits which are capable of discriminating against noise and switch precisely at zero crossing are not necessary when used in conjunction with a COPS device. COPS software is capable of compensating for noisy or semi-accurate zero voltage detection circuits. This can be accomplished by introducing delays and debounce techniques in the software routines. With a given reference point in the AC waveform it now becomes easy

to divide the waveform to efficiently allocate processing time. These techniques are illustrated in the code listing at the end of this paper.



TL/DD/6939-5

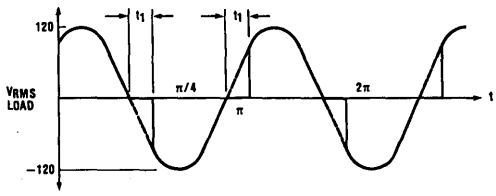
FIGURE 5. Current Lag Caused by Inductive Load, Snubber Circuit

2.2 PROCESSING TIME ALLOCATIONS

Half Cycle Approach

In order to accomplish more than triac timing, dead delay time must be turned into computation time. It appears that the controller is occupied totally by time delays, which leaves a very limited amount of additional control capability. There are, however, many ways to accomplish auxiliary tasks simultaneously.

On each half cycle an initial delay is incorporated to space into the cycle. This dead time may be put to use and very little voltage to the load is sacrificed. For example, if the load is switched on at $\pi/4$ RAD, the maximum applied RMS voltage to the load is $114V_{RMS}$ (assuming V_{SUPPLY} = $120V_{RMS}$). This is illustrated in the figure below.



TL/DD/6939-6

FIGURE 6. Full Cycle Approach

If a delay of $\pi/4$ RAD (45 degrees) is inserted after each zero crossing detection the RMS voltage to the load can be determined in the following manner:

$$V_{LOAD} = \sqrt{\frac{(120\sqrt{2})^2}{(2)\pi}(2)\int_{\pi/4}^{\pi} \sin^2(a)\,da}$$

$$V_{LOAD} = \sqrt{\frac{(120\sqrt{2})^2}{(2)\pi}(2)(1.428)}$$

$$V_{LOAD} = 114.4\ V_{RMS}$$

$\pi/4$ RAD = 45 degrees @60 Hz t = 2.08 ms

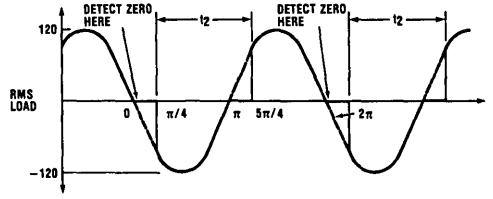
As can be seen the dead time on each half cycle can be 2.08 ms and the load will still see 114.4 V_{RMS} of a V_{SUPPLY} of 120 V_{RMS}. If this approach is implemented the initial delay of 2.08 ms can be used as computation time. The number of instructions which can be executed when operating at 4 μs instruction cycle time is:

2.08 ms/4 μs = 520 instructions

(130 instructions at 16 μs cycle time)

Full Cycle Approach

The methods of half cycle and full cycle triggering are very similar in procedure. The main difference is that all timing is referenced from only one (of the two) zero voltage detection transition in each full AC cycle. For most all applications, when varying the conduction angle it is desirable to fire at the same conduction angle each half cycle to maintain a symmetric applied voltage. In order to accomplish this the triac may be fired twice from one reference point. When applying this technique an 8.33 ms delay must be executed to maintain the symmetric applied voltage. This approach provides the most auxiliary computation time in that the 8.33 ms delay may be turned into computational time. The basic flow for this technique is illustrated below.



TL/DD/6939-7

FIGURE 7. Full Cycle Approach

In the above example the zero crossing pulse is debounced on the one-to-zero transition, thus marking the beginning of a full cycle. Once this transition has been detected, an ini-

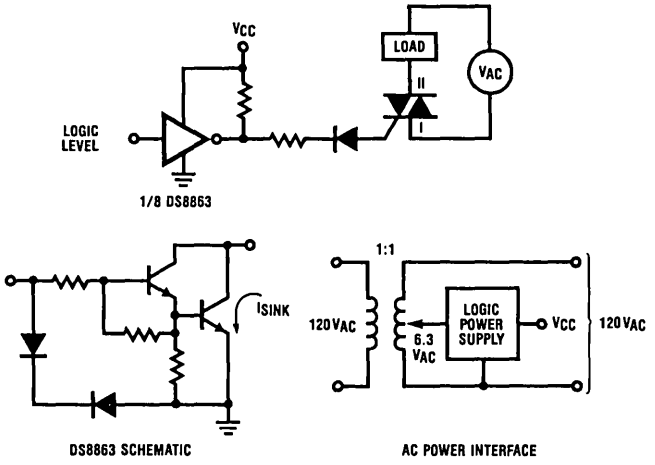


FIGURE 8. Steady State Triggering

tial delay of $\pi/4$ RAD is incorporated and the triac is fired. At this time exactly 8.33 ms is available until the triac need be triggered again. This will provide a symmetric voltage to the load only if the delay is 8.33 ms. During this period the number of instructions which can be executed when operating at 4 μs is:

$$8.33 \text{ ms}/4\ \mu\text{s} = 2082$$

(520 instructions at 16 μs)

An alternative approach may be to take the burden from the COPS device by using peripheral devices such as static display controllers, external latches, etc.

2.3 STEADY STATE TRIGGERING

It is possible to trigger a triac with a steady state logic level. This is accomplished by allowing the triac gate to sink or source current during the desired on-time. When utilizing this method it becomes easier to trigger the triac and leave it on for many cycles without having to execute code to retrigger. This approach is advantageous when the triac must be fired is for relatively long periods and conduction angle firing is not desired, thus more time is available to accomplish auxiliary tasks. A steady state on or off signal and external circuitry can accomplish triac firing and free the processor for other tasks. If it is desired to use a pulse transformer, an external oscillator must be gated to the triac to provide the trigger signal. A pulse train of 10 to 15 kHz is adequate to fire the triac each half cycle. This calls for external components and is relatively costly. If isolation associated problems can be tolerated or overcome (dual power supply transformers, direct AC coupling, etc.), a simple buffer may be utilized in triggering the triac. This method is illustrated in *Figure 8*. The National Semiconductor DS8863 display driver is capable of steady state firing of the triac. National offers many buffers capable of driving several hundred milliamps, which are suitable for driving triacs. On the market today there are many suppliers of sensitive gate triacs which may be triggered directly from a COPS device or in conjunction with a smaller external buffer.

The DS8863 display driver is capable of sinking up to 500 mA, which is adequate to drive a standard triac. In the off state the driver will not sink current. When a logic "1" is applied to the input the device will turn on. Keeping the device off (output "1") will prevent the triac from turning on because the buffer does not have the capability of sourcing current. A series resistor limits the current from the triac gate and the diode isolates the negative spikes from the gate. Since the drive circuit will only sink current in this configuration, the triac will be operating in the I- and III- modes.

3.0 Triac Light Intensity Control Code

The following code is not intended to be a final functional program. In order to utilize this program, modifications must be made to specialize the routines. This is intended to illustrate the method and is void of control code to command a response such as intensify or deintensify. The control is up to the user and full understanding of the program must be attained before modifications can be implemented.

This program is a general purpose light intensifying routine which may be modified to suit light dimmer applications. The delay routines require a 4.469 μs cycle time which can be attained with a 3.578 MHz crystal (CKI/16 option). This program divides the half cycle of a 60 Hz power line into 16 levels. Intensity is varied by increasing or decreasing the conduction angle by firing the triac at various levels. The program will increase the conduction angle to a maximum specified intensity in a fixed amount of time. The time required to intensify to the maximum level is dependent on the number of fire-times per level that is specified (FINO). This code illustrates a half cycle approach and relies on the parameters specified by the programmer in the control selection.

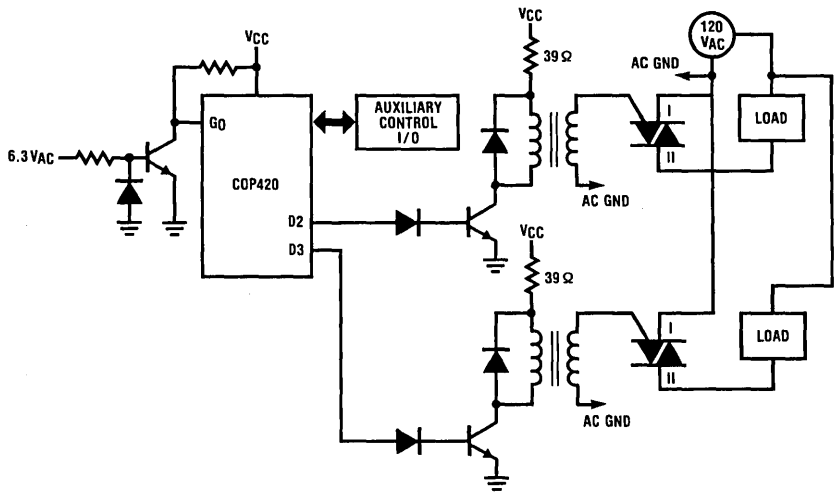
Zero crossings of the 60 Hz line are detected and software debounced to initiate each half cycle; thus the triac is serviced on every half cycle of the power line. A level/sublevel approach is utilized to vary the conduction angle and provide a prolonged intensifying period. The maximum intensity is specified by the "LEVEL" RAM location and time required to get to that level is specified by the "FINO" RAM location.

Once a level has been specified, the remaining time in the half cycle is then divided into sublevels. The sublevels are increased in steps to the maximum level. The "FINO" RAM location contains the number of times that the triac will be fired per sublevel, thus creating the intensity time base. There are 15 valid sublevels and up to 15 fire-times per sublevel. Both these parameters may be increased to provide better resolution and longer intensify periods. To make the triac de-intensity (dim) the sublevels need only to be decremented rather than incremented. If this is done, the conduction angle will start out at the maximum level and dim by means of stepping down the sublevels. When modifying this routine to incorporate more resolution or increased versatility, care must be taken to account for transfer of control instructions to and from the delay routines.

The following is a schematic diagram of the COPS interface to 120V_{AC} lamps. The program will intensify or de-intensify the lamps under program control.

3.1 TRIAC LIGHT INTENSIFY ROUTINE

This program intensifies a light source by varying the conduction angle applied to the load. The maximum level of intensity is stored in "LEVEL," and the time to get to that level is specified by "FIND." Both these parameters may be altered to suit specific applications. To cause the program to de-intensify the light source, the sublevels must be decremented rather than incremented.



TL/DD/6939–9

FIGURE 9. Triac Interface for COPS Program

```
; TRIAC LIGHT INTENSIFY ROUTINE
;
;
; THIS PROGRAM INTENSIFIES A LIGHT SOURCE BY VARYING THE
; CONDUCTION ANGLE APPLIED TO THE LOAD. THE MAX LEVEL
; OF INTENSITY IS STORED IN 'LEVEL' AND THE TIME TO GET TO
; THAT LEVEL IS SPECIFIED BY 'FIND'. BOTH THESE PARAMETERS
; MAY BE ALTERED TO SUIT SPECIFIC APPLICATIONS. TO CAUSE
; THE PROGRAM TO DE-INTENSIFY THE LIGHT SOURCE, THE
; SUBLEVELS MUST BE DECREMENTED RATHER THAN
; INCREMENTED.
;
;
        TEMP1   =1,0    ; TEMPORARY DELAY COUNTER
        FIND    =0,9    ; NUMBER OF FIRE TIMES
        LEVEL   =0,0    ; MAX LEVEL
        SUBLEV  =1,10   ; SUBLEVEL COUNT
        TEMP    =1,11   ; TEMPORARY DELAY COUNTER
;
; HERE THE OPERATING PARAMETERS ARE DEFINED AND LEVEL
; INITIATION IS SPECIFIED
;
        .FORM
        .PAGE   0
        CLRA            ; REQUIRED
CLRAM:  LBI     3,15    ; ROUTINE TO CLEAR ALL RAM
CLR;    CLRA
        XDS
        JP      CLR
        XABR
        AISC    15
        JP      BEGG
        XABR
        JP      CLR
;
; THIS SECTION INITIATES CONTROL ON POWER UP OR RESET
; AND SYNCHRONIZES THE COPS DEVICE TO THE 60 HZ AC LINE
;
BEGG:   OGI     15      ; OUTPUT 15 TO G PORTS TO PULL
                        ; UP ZERO CROSSER INPUT
        LBI     LEVEL   ; SPECIFY MAX LEVEL
        STII    7
        JSR     OUT     ; COPY TO TEMP1
BEG:    SKGBZ   0       ; SYNC UP TO 60 HZ
        JP      HI      ; READY NOW
        JP      BEG     ; WAIT TILL G IS 1
;
; THIS SECTION PROVIDES THE DEBOUNCE FOR THE ZERO
; VOLTAGE DETECTION INPUT AND COMPENSATES FOR THE
; OFFSET OF THE DETECTION CIRCUIT
;
HI:     SKGBZ   0       ; TEST GO FOR ZERO CROSS
        JP      HI      ; HIGH LEVEL
; GETS HERE ON FIRST TRANSITION
        CLRA            ; START OF DEBOUNCE DELAY
        AISC    1
        JP      .-1
; DID A LITTLE DELAY, IS IT STILL 0
        SKGBZ   0       ; TEST FOR 0
        JP      HI      ; FALSE ALARM
; MUST HAVE HAD SOME NOISE GO BACK AND WAIT FOR TRUE ZC
DOIT:   JMP     INT     ; VALID TRANSITION, SERVICE
                        ; TRIAC
LO:     SKGBZ   0       ; DEBOUNCE IN 0 TO 1
        JP      DDD     ; MAY HAVE SOMETHING THERE
        JP      LO      ; NO WAIT HERE FOR A BIT
DDD:    CLRA            ; GOING TO WAIT AND SEE
        AISC    1
        JP      .-1
        SKGBZ   0       ; WELL, DO WE HAVE A CLEAN
                        ; TRANSITION
        JP      DELL    ; YES, GO TO MAIN ROUTINE
        JP      LO      ; FALSE ALARM, TRY AGAIN
DELL:   CLRA            ; DO A DELAY TO COMPENSATE
DEL:    NOP
        NOP             ; FOR NON SYMMETRIC ZC
        NOP
        AISC
        JP      DEL     ; KEEP DELAY GOING
        JP      DOIT    ; GO TO MAIN ROUTINE

        .FORM
        .PAGE   1
;
;
; THIS IS THE MAIN ROUTINE FOR THE INTENSIFY/DE-INTENSIFY
; OPERATIONS. TRANSFER OF CONTROL TO THIS SECTION
; OCCURS AFTER ZERO VOLTAGE CROSSING EACH HALF CYCLE.
; THIS MAKE USE OF TEMP REGISTERS THUS PARAMETERS
; NEED NOT BE REDEFINED FOR EACH OPERATION.
;
;
INT:    CLRA
        ADT             ; DELAY INTO WAVEFORM
        LBI     TEMP    ; USE TEMP REG
        X
        JSRP    PORT    ; DO DELAY
POINT:  LDD     LEVEL   ; POINT TO LEVEL TO INITIATE
                        ; DELAY
                        ; DELAY TO MAX LEVEL
        XAD     TEMP    ; USE TEMP DIGIT TO DELAY
TAMP:   LBI     TEMP
        LD
        AISC    15      ; ARE WE AT THE LEVEL ?
        JP      ATLEV   ; MADE IT TO THE LEVEL
        X               ; NO
        JSRP    DE5     ; DO SERIES OF .5MS TO GET
                        ; THERE
        JP      TAMP    ; KEEP DOING IT
ATLEV:  LDD     SUBLEV  ; AT MAX FIRE LEVEL
        XAD     TEMP    ; INIT FOR SUBLEVEL DELAY
JK:     LBI     TEMP
        LD
        AISC    1       ; AT SUB LEVEL ?
        JP      TRE     ; NO DO DELAY
        JP      SBLEV   ; YES
TRE:    X
        JSRP    SPDL    ; VARIABLE DELAY
        JP      JK
SBLEV:  LBI     FIND
        JSRP    DEC     ; DEC FIRE NUMBER
        AISC    1       ; TEST IF FIND AT 15
MAXLEV: JMP     FIRE    ; NO KEEP FIRING AT THAT LEVEL
        LBI     SUBLEV  ; YES INC SUBLEVEL
        CLRA
        AISC    14      ; IS MAX SUBLEV REACHED
        SKE
        JP      THERE   ; NO INC SUBLEV
        JP      MAXLEV  ; YES FIRE IT
THERE:  JSRP    INC     ; GO TO NEXT SUBLEVEL
        LBI     FIND
        STII    14      ; SET FIRE TIME
        JP      MAXLEV  ; GO FIRE

        .FORM
        .PAGE   2
```

```
; SUBROUTINE PAGE
INC:    CLRA
        AISC    1
        JP      ADEX        ; GO ADD ONE TO DIGIT
DEC:    CLRA                ; 0 TO A
        COMP                ; CREATE A 15
ADEX:   ADD                 ; ADD A TO RAM
        X                   ; PUT BACK (D - 1 IN A NOW)
        RET
DE5:    LBI     0,10        ; DELAY ROUTINE
        CLRA                ; WILL BE REPLACED LATER
        AISC    3
        JP      .-1
        LD
        XIS
        JP      .-5
        RET                 ; DONE DELAY
FIRE:   LBI     0,15        ; PULSE D OUTPUT
        OBD
        NOP
        NOP
        NOP
        LBI     0,0
        OBD
        SKBGZ   0           ; TEST WHICH DEBOUNCE IS
                            ; NEEDED
        JMP     HI          ; DEBOUNCE ONE TO ZERO
        JMP     LO          ; DEBOUNCE ZERO TO ONE
SPDL:   LBI     TEMP1       ; TEMP1 IS A TEMP REG
PORT:   LD                  ; VALUE IN TEMP1 DICTATES
        AISC    1           ; THE AMOUNT OF DELAY
        JP      FOY
OUT:    LBI     LEVEL       ; ALSO USED TO COPY LEVEL
        LD      1           ; RESTORE LEVEL
        X
        RET
FOY:    X
        JP      PORT
        .END
```

Testing of COPS™ Chips

National Semiconductor
COP Note 7

Table of Contents

This note will provide some insight into the test mode, the mechanics of testing, and the philosophy of how to implement a test of the COP-400 microcontrollers. Other than the obvious, (verifying that the part meets the specifications), the reason for the test must be considered. Somewhat different criteria may hold, depending on the objective. The manufacturer wafer sort or final test can differ from an incoming inspection at the user's plant, or a field reject test. The first two tests have limited interest as this is not a justification of the testing done on the part during manufacture. Rather, this is a guide for those doing user functional testing.

1.0 INTRODUCTION

Since the introduction of the very first semiconductor devices, testing has been a major problem and expense in their production and use. As the complexity has risen, testing has become a more significant factor. With today's single chip microcontrollers like the COPS devices this is particularly true as one has a complete computer system in a chip. In order to reduce the testing burden, the facilities to ease the testing have been built into the COPS devices. With the test ability built into the device for production test, the user need only follow set procedures to verify the chip at incoming inspection or field test.

2.0 PHILOSOPHY

The basic test philosophy requires that four major areas be exercised. These areas are:

1) Synchronize the device and tester.
2) Test the internal logic and I/O.
3) Test the RAM.
4) Verify the ROM program.

If the devices perform all of these four properly, the device is good. This is a reasonable assumption with a standard device that has a debugged test routine and is ROM programmed. A custom circuit just going into production might not have the accumulated test background. By attacking the problem on a "sum of the parts" approach, one need not do any exhaustive functional test on routine production parts. This will be a major gain where lengthy time consuming or time dependent routines are involved. If one attempts to do a functional test of the chip, a sequence that is unique to the application is needed. Thus, a test program must be written and debugged for each ROM pattern. Further, a test box/board must be designed, built, debugged, documented, and maintained for each one. If testing has been considered from the beginning, the chip will have built-in capabilities to exercise the various parts of it. The different functional parts and instructions are tested to verify proper operation at the voltage and frequency limits.

3.0 BUILT-IN TEST FEATURES

The first step in testing the COP400 devices is to understand the built-in test control features. This will involve the SI/O and the L lines. The SO pin has been designed to be the control node for testing. The pin will normally be in an active low state and when forced high externally, places the chip in the test mode. It should be noted that this output can sink considerable current and one should not force the pin to the V_{CC} rail. By limiting the voltage to the 2.0/3.0V range one can not damage the device where the application of a higher voltage could. When forced into the test mode the SI pin controls the sub mode of the chip. With SI high the data placed on the L port is used as an instruction. When SI is low (and the L output is enabled) the contents of the ROM will be dumped out through the L port. Certain other internal functions have been implemented to allow these modes but these are not part of the basic operation. Included in this category is the activation of the skip signal to prevent the program counter from jumping out of sequence by executing a program control instruction.

3.1 Sync Between Tester and DUT

In order to be able to test a COPS chip, the tester must be in sync with the device under test (DUT). By using an external oscillator the two may be run at the same frequency. This is true regardless of the option or type of oscillator chosen for the chip. Even the RC configuration may be overridden with an external signal that meets the level requirements. In addition to running at the same frequency, the chip and tester must be in sync on a bit basis. See *Figure 1.* The supportive features mentioned above include the condition of the SK signal being a bit (instruction) clock until stopped by software in the program. Hence, one can start the tests based on an edge change of SK. It is important that this be accurate because all data I/O changes will be relative to the SK timing (see the appropriate device data sheet).

It should also be noted that the oscillator frequency is programmed to a rate of 4-32 higher than SK. If one is building a test fixture for more than one device, some method must be available to enter this number. If one is testing a COP420 or COP421 near its upper limit it would be wise to do the SK sync operation at a lower rate and then increase the input frequency. This is desirable because the phase relationship is close to TTL propagation delays at the upper limit. Implementation of the area could be a preset counter that is gated on after a zero to one transition is seen on SK. Continual comparison could be made but once in sync, there should not be any need for the comparison as they should remain in sync.

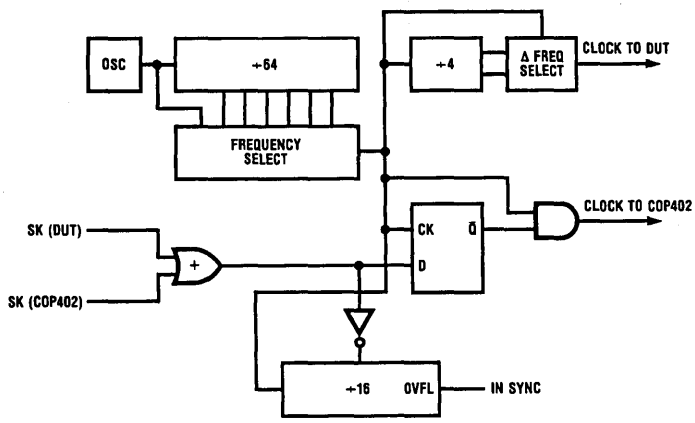
The basic use of this "sync counter" is to derive the proper timing for loading data and instructions into the chip and verify the outputs. The COP402 data sheet should be used as a guide for these times, modified properly for the L and C parts. For those designing testers, it is suggested that one not attempt to test worse case timing changes as these could be very difficult to implement. Like other parametric tests these should in general be left to the professional test equipment.

3.2 Internal Logic Test

With the device and the tester in sync, actual testing may begin. See the sequence control circuit of *Figure 2.* To place the chip into the test mode the SO output is pulled to a one level (between 2.0 and 3.0 volts). It should be pulled with a circuit that will limit the upper voltage to 3V as this output can have a significant current sink capability. On power up (or after reset) the SO line is set to a zero by the internal logic. An internal sense line will detect the forced condition and provide test control. A delay of 10 ms should be taken after power-up to allow the power on reset circuit to time out before instructions can be executed. If the reset pin is activated in mid-program for some reason, several instructions cycle times should be ignored to insure complete operation.

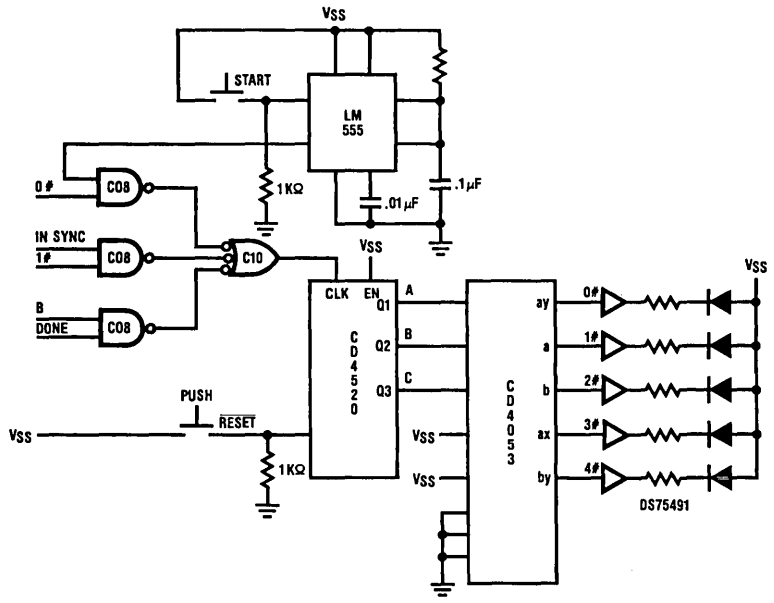
The tester should at this point force instructions into the L port. These instructions will be executed as if they were from the ROM. The sequence of the instructions is not particularly critical. Table I gives an example sequence. The main steps are to be able to detect an output change (OGI) early to verify connection/operation. It is much better to find a problem before going through the steps of loading RAM and then finding that the chip doesn't work. All instructions should be exercised although certain ones should be postponed. Enabling the Q register to the L port is an example. This would interfere with the insertion of instructions on the L port. Another problem is the SO test which could be set up with an XAS and then released from the test mode to check proper data output.

Certain commands will require more effort than others. To check the program counter during JMP's and sub-routine operation will require that known info at the new address be available. One should execute a JSRP at some known address and release the test mode to see that the operation in the subroutine (e.g., SC) is done and that a return is made to N + 1. At this point test mode can be re-established to continue the test. The main point to remember is to provide a positive indication of the success of that specific test.



TL/DD/6940–1

FIGURE 1. Tester Clock Generation and Synchronization Circuit



TL/DD/6940-2

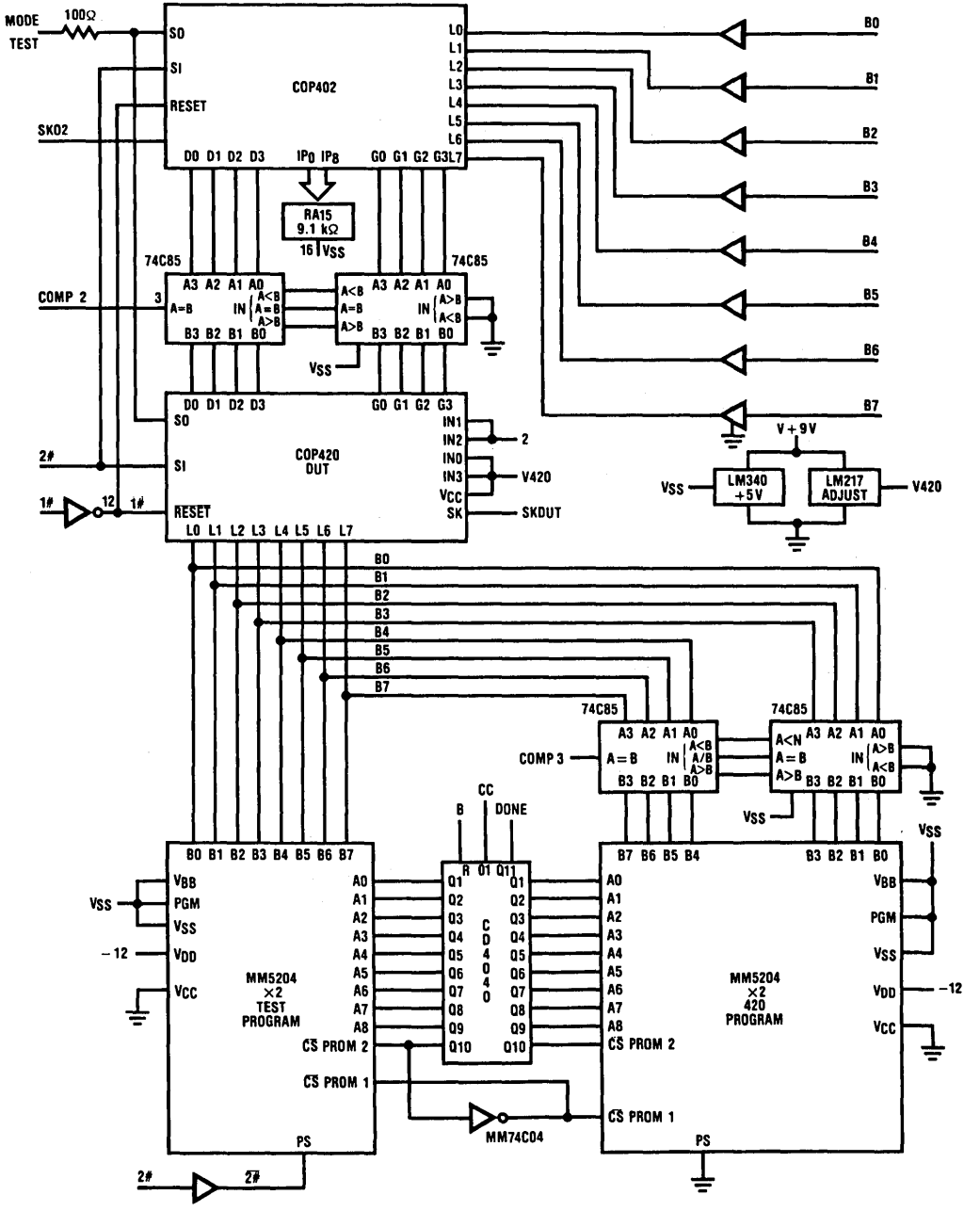
FIGURE 2. Tester Mode Sequencer

3.3 RAM Test

The verification of RAM is a part of the internal logic test, but is treated separately here. One must check both the RAM and its address register to find all faults. An example of this testing would be to load RAM with a string of STII commands. By then going back and reading this data to the outside (through an OMG instruction in a loop) the tester could verify both RAM and address were functional. One could then load RAM with all 6's and 9's (or 5's and 10's) sequentially to insure that all bits were functional and adjacent bits not shorted. Other similar tests could be run at the discretion of the user to do further testing. All of these tests would utilize the output of data via the G ports to validate the data. See the comparator circuit *Figure 3.*

3.4 ROM Dump

Successful operation of the internal logic tests and RAM will lead to the final test phase, ROM comparison. In order to check the ROM contents, the ROM dump mode must be entered. One should force a JMP to an address near the end of the ROM space (3FF for a 420 chip, 1FF for a 410). A desirable point might be 3FA. The program counter will step ahead on each instruction cycle unless a program control is executed. The next step is to load the Q register with a non-conflicting value so that the enabling of the L outputs will not destroy the second byte of the LEI instruction as control is passed into the ROM dump mode. After going to this address, one should execute an enable of the L lines to the output port (LEI 4). Having done this the external buffers should be disabled and the SI pin taken low. This will allow data out and remove potential level conflicts. By letting the PC step ahead to address zero one can then begin the byte by byte comparison of data. In this mode the controller is not executing the code because the skip line is enabled throughout the sequence. By halting a counter on a failure, one could determine the questionable address.



TL/ DD/6940–3

FIGURE 3. Functional Logic and RAM Comparison Circuit

TABLE I. Typical Test Sequence

INSTRUCTION	RESULT	COMMENTS
NOP	NO CHANGE	CHECK NOP & ALLOW TRANSIENT
		CYCLE FOR MODE
OGI 9	G(0 > 9)	NOT ON 410L/411L
OGI 6	G(9 > 6)	REVERSE ALL G STATES
STII 8		SET UP 0,0 FOR FUTURE
LBI 3,13		B TO NEW POSITION (3, 13)
OBD	D(0 > 13	CHECK D
CLRA		MAKE SURE A = 0
XABR		3 > A; 0 > Br
CAB		MOVE 3 to Bd
OBD	D(13 > 3)	CHECK XABR CAB & D CHANGE
CLRA		!
AISC 2		!FORCE A > 2
CAB		2 > Bd
OBD	D(3 > 2)	VERIFY 2 FROM A > Bd
STII 7		7 > 0.2 & Bd > 3
OBD	D(2 > 3)	STII INCREMENTS Bd
CAB		SEE THAT A STILL THE SAME
OMG	G(6 > 7)	OMB & RAM CHECK
CLRA		
CAB		B(0,0)
OMG	G(7 > 8)	TIE IN RAM, A & G OPERATION
SMB 0		SMB INST. CHECK
OMG	G(8 > 9)	:
SMB 1		:
OMG	G(9 > 11)	:
RMB 0		:
RMB 3		:
X		:0 > 0,0;2 > A
CAB		A = 2 > B
OMG	G(11 > 7)	OUTPUT M(0,2)
LD 1		M(0,2) > A; B > 1,2
XAD 0,0		A(7) < -> M(0,0) 2
AISC 15		AISC CHECK; A = 1
LDD 0,0		CHECK SKIP OF 2 BYTE INST.
X		STORE 1
OMB	G(7 > 1)	VERIFY
LD 0		COPY1,2 BACK TO A
ADT		ADD TEN
XDS		LEAVE 11 IN 1,2;GO 1, 1 WITH 1
XDS		LEAVE 1 IN 1,1;GO 1,0 W ?
OBD	D(2 > 0)	CHECK Bd MOVEMENT
STII 5		5 > 1,0;Bd TO 1,1
CBA		CHECK B > A
AISC 3		AISC CHECK 4 >A

INSTRUCTION	RESULT	COMMENTS
XDS		1 > A; 4 > 1,1
OMG	G(1 > 5)	FROM 1,0
XDS		5 > A; 1 > 1,0; Bd < 15 SKIP
LDD 0,0		SKIPPED !
OBD	D(0 > 15)	
AISC 4		9 > A
X		9 > 15
OMG	G(5> 9)	
CLRA		
COMP		ONES TO A
XOR		FLIP MEMORY
XIS		6 > 1,15; 9 > A; Bd > 1,0
LDD 0,0		SKIP
SKE		
LB 1,2		SKIP 2 WORD LBI (NOT IN 410)
OBD	D(15 > 0)	VERIFY WORD
SKE		11 NOT = 9
LBI 1,0		BACK TO 1,0
SMB 2		:
SKE		:
RMB 2		:
SKE		:CHECK BIT
SMB 3		:MANIPULATIONS
SKE		:
LDD 0,0		:
X 3		Bd > 2,0
XAD 1,1		9 > 1,1; 4 > A
XIS 1		4 > 2,0; Bd > 3,1
ING		INPUT G PORT
X		STORE

INSTRUCTION	RESULT	COMMENTS
CLRA		
ASC		CHECK ADD WITH CARRY
SC		CHECK SET CARRY
SKC		CHECK SKIP ON CARRY
LDD 0,0		
X		STORE A
OMG	G = 9	NO CHANGE
CLRA		
ASC		
X		
OMG	G(9 > 10)	CARRY ADDS ONE TO MEMORY
CAMQ		STORE A & M IN Q; 10,9
XDS		9 > 3,1; 10 > A; Bd > 3,0
X		STORE 9 IN 3,0
OMG	G(10 > 9)	
LD 2		9 > A; Bd > 1,0

INSTRUCTION	RESULT	COMMENTS
OMG	G(9 > 1)	
LD 3		1 > A; Bd > 2,0
OMG	G(1 > 2)	
ADD		ADD WITHOUT CARRY
X		STORE 3 IN 2,0
SC		
LDD 0,0		7 > A
CASC		CHECK CASC
SKC		
X		STORE 12
OMG	G(2 > 12)	
CLRA		:
AISC 3		:
X		:
SC		:CHECK
SKC		:SKC/SC
X		:
OMG	G(12 > 3)	
RC		:
SKC		:CHECK
X		:RC
OMG	G(3 > 12)	:
LBI 0,0		:CHECK
LBI 1,15		;SEQUENTIAL LBI'S
LBI 2,7		ALSO SKIPPED (LBI 2,7 NOT IN 410)
OMG	G(2 > 7)	
CQMA		LOAD CONSTANTS FROM Q
OMG	G(7 > 9)	CHECK
X		:
OMG	G(9 > 10)	:
LEI 1		
XAS		STORE A — > S (9)
CLRA		
AISC 7		:
SKGBZ 0		:
X		:CHECK
OMG		:
SKGBZ 1		:
X		;G BIT
OMG	G (10 > 7)	:
SKGBZ 2		:
X		:
OMG	G(7 > 10)	:TESTS
SKGBZ 3		:
X		:
OMG	G(10 > 7)	:

INSTRUCTION	RESULT	COMMENTS
SKGZ		
X		:CHECK
OMG	G(7 > 10)	:
OGI 0	G(10 > 0)	:G TEST
SKGZ		:
X		:
OMG	G(0 > 10)	:
SKMBZ 0		
X		CHECK MEMORY BIT TESTS
OMG		NO CHANGE
SKMBZ 1		

TABLE I. Typical Test Sequence (Continued)

INSTRUCTION	RESULT	COMMENTS
X		
OMG	G(10 > 7)	NO SKIP
SKMBZ 2		
X		WON'T SKIP
OMG	G(7 > 10)	
INIL		SEE THAT L LATCHES RESET
ININ		ASSUME G – > I
SKE		
X1		Br > 1
OMG		SHOULD BE EQUAL
INIL		:
X		:
SKMBZ 3		:
OBD	D(15 > 0)	:INIL TEST
OGI 1		:
LBI 3,11		:
OGI 0		:
INIL		:
X		:
SKMBZ 0		:
OBD	D(0 > 11)	:
NOP		
XAS		:
X		:XAS TEST
OMG	G(10 > 9)	:

INSTRUCTION	RESULT	COMMENTS
LBI 0,0		LOAD RAM WITH
STII 7		CONSTANTS USING
STII 14		STII
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		
LBI 1,0		
STII 7		
STII 14		
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		
LBI 2,0		
STII 7		
STII 14		
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		

INSTRUCTION	RESULT	COMMENTS
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		
LBI 3,0		
STII 7		
STII 14		
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		

INSTRUCTION	RESULT	COMMENTS
LBI 0,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
LBI 1,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

TABLE I. Typical Test Sequence (Continued)

INSTRUCTION	RESULT	COMMENTS
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
LBI 1,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
LBI 3,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
JMP X	INITIALIZE—SELECT ADDRESS X FOR OGI OR OMG (SELECT LBI FOR KNOWN DATA)	
RELEASE TEST MODE	OBD (SELECT B FOR KNOWN CONDITION) CHECKS JMP	

TABLE I. Typical Test Sequence (Continued)

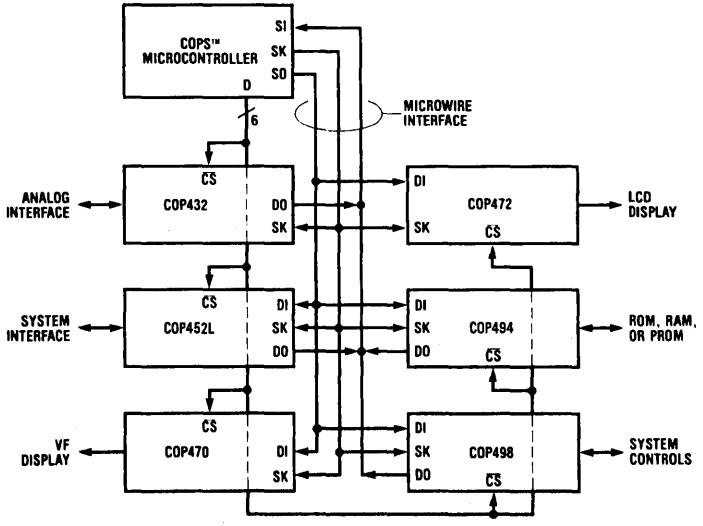
INSTRUCTION	RESULT COMMENTS
SET TEST MODE	
JP X-2	:
JSR Y	CHECK JP & JSR
RELEASE TEST MODE	"Y" SHOULD CHANGE THE OUTPUT CONDITIONS OF "X"
EXECUTE CODE (Y)	IF AT ALL POSSIBLE
SET TEST MODE	
RET	
RELEASE TEST MODE	
EXECUTE "X" AGAIN	VERIFIES RET
SET TEST MODE	
JP X-2	
JSRP Z	CHECK JSRP & RETSK
RELEASE TEST MODE	
EXECUTE CODE	"Z" SHOULD CHANGE "X" OUTPUT CONDITIONS
SET TEST MODE	
RETSK	DON'T CHANGE Z CONDITIONS — RETSK
RELEASE TEST MODE	
EXECUTE	" "
SET TEST MODE	
LOAD A & M TO	FIND VALUE OF ADDRESS IN BLOCK (4 PAGES)
VALUE OF ADDRESS	AT OR JUST BEFORE AN OUTPUT
TO GO TO	CHANGE SET A & M TO ADDRESS
OUTPUT CHANGE	OF "VALUE"
JID	CHECKS JID
RELEASE TEST MODE	
EXECUTE OUTPUT	
SET TEST MODE	LOAD A & M WITH A UNIQUE ADDRESS
LOAD A & M	SUCH THAT CONTENTS OF THAT ADDRESS WILL BE SEEN ON G
LQID	
X064	;OR USE THIS CAUSE THE DATA COMES ;FROM YOUR TESTER ANYWAY
CQMA	
OMG	LQUID & CQMA CHECKED
X	
OMG	"
INL	:
OMG	G − > 2 INL TEST (COPY OF 2nd BYTE)
X	
OMG	G − > E :

This test sequence is not to be taken as a recommended test routine and is only shown as an example of what might be done to test various COPS parts. It is also advisable to approach measurements in the test mode with some caution. As stated earlier, one can force a large current into the SO node to place the chip in the test mode. Not only can this current do damage if unlimited, but it can also cause local current overloading such that some I/O conditions may be adversely affected. Obviously this will be more pronounced at higher V_{CC} voltages. A specific example is that the L output current sink test should only be tested at a V_{OUT} of 0.4V and 0.36 mA as the more stringent tests can exceed power limits when combined with the SO current.

MICROWIRE™

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and/or may permit the COPS controller to be packaged in a smaller (and even lower cost) package. (MICROWIRE peripherals may also be used with non-COPS controllers). For further applications information, refer to COPS Briefs 8 and 9. MICROWIRE makes sense.

The example below illustrates the power and versatility of MICROWIRE via an extreme example—using one of each type of peripheral with a single controller.



TL/DD/6940–4

COP431 SERIES, 8-BIT A/D CONVERTERS

The COP431 series is an 8-bit successive approximation A/D converter with a serial I/O and configurable input multiplexer with up to 8 channels. The serial I/O is configured to comply with the NSC MICROWIRE serial data exchange standard for easy interface to the COPS family of processors, and can interface with standard shift registers or other μPs.

The 2, 4 or 8 channel multiplexers are software configured for single-ended or differential inputs as well as channel assignment.

The differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

COP452L FREQUENCY/COUNTER PERIPHERAL

The COP452L contains 2 independent 16-bit counter/register pairs, and is well suited to a wide variety of tasks involving the measurement and/or generation of times and/or frequencies. Included are multiple tones, precise duty cycles, event counting, waveform measurement, "white noise" generation, and A-D/D-A conversions. An on-chip zero-crossing detector can trigger a pulse with a programmed delay and duration.

COP470 V.F. DISPLAY DRIVER

The COP470 is designed to directly drive a multiplexed Vacuum Fluorescent display. Data is loaded serially and held in internal latches. The COP470 has an on-chip oscillator to multiplex four digits of eight segment display, and may be cascaded and/or stacked to drive more digits, more segments, or both.

With the addition of external drivers, the COP470 also provides a convenient means of interfacing to a large-digit LED display.

COP472-3 LIQUID CRYSTAL DISPLAY CONTROLLER

The COP472-3 Liquid Crystal Display (LCD) Controller drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472-3 contains an on-chip oscillator and generates all the multilevel waveforms for backplanes and segment outputs on a triplex display. One COP472-3 can drive 36 segments multiplexed as 3 × 12 (4½ digit display). Two COP472-3 devices can be used together to drive 72 segments (3 × 24) which could be an 8½ digit display.

COP494 256-BIT SERIAL ELECTRICALLY ERASABLE PROGRAMMABLE MEMORY

The COP494 is a 256-bit non-volatile memory. The device contains 256 bits of read/write memory divided into 16 registers of 16 bits each. Each register is serially read or written by the COP400 Family Controller. Written information is stored in a floating gate cell with at least 10 years of retention.

COP498/COP499 LOW POWER CMOS RAM AND TIMER

The COP498 low power CMOS Random-Access Memory and Timer is an external memory and timer chip with the simple MICROWIRE serial interface. The device contains 256 bits of read/write memory divided into 4 registers of 64 bits each. The COP498 also contains a crystal-based timer for timekeeping purposes, and can provide a "wake-up" signal to turn on a COPS controller.

The COP498 can be used for low power standby memory and can also be used for low power operation by turning the controller off and on, on a duty cycle basis.

The COP499 Low Power CMOS Random-Access Memory is an external memory and switch chip with the simple MICROWIRE serial interface. The device contains 256 bits of read/write memory divided into 4 registers of 64 bits each. The COP499 also contains circuitry that enables the user to turn a controller on and off while maintaining the integrity of the memory.

Current Consumption in NMOS COPS™ Microcontrollers

National Semiconductor
Application Brief 3
Len Distaso

Current consumption in the N-channel COPS microcontrollers is a function of manufacturing process variation and three operating condition parameters: temperature, voltage, and frequency. The aforementioned process variation swamps all other variations. Of the operating condition parameters, temperature is by far the most significant. This application brief is intended to provide the user with a guide to approximate the worst-case current consumption of the NMOS COPS microcontroller at a given set of operating conditions and to approximate the current variation with respect to temperature, voltage, and frequency.

Note that this is a guide only. Some approximations in the equations have been made. Only the current values found in the various device data sheets are guaranteed. Values derived by the techniques described here are neither guaranteed nor tested.

PROCESS VARIATION

If a user were to measure the current in two identical COPS microcontrollers under identical operating conditions (i.e., same temperature, voltage, and frequency), the results would probably be different. The reason for this difference is variation in the manufacturing process within its valid range. This variation can be quite substantial; a range of about 3 to 1 can be expected. This variation is essentially a device-to-device variation and basically not related to the operating conditions of the device. The three operating condition parameters (temperature, voltage, and frequency) affect current in the manner described below.

The values for current consumption in the various device data sheets are worst-case maximum values and assume that the processing parameters are at the end of the valid range which will produce maximum current consumption in the device.

THE EFFECT OF FREQUENCY

The frequency effect on current consumption is primarily a device design consideration. The higher the intended operating frequency, the higher the maximum current. However, once the device is designed in this process for a given maximum frequency, there is little variation with operating frequency. To be sure, there is some variation. As might be expected, current consumption is greater at higher frequencies. The variation is, however, slight—typically less than 5%.

THE EFFECT OF VOLTAGE

The operating voltage of the microcontroller has a slightly greater effect on current consumption than the operating current. Current consumption increases with increasing operating voltage. On examining the MOS device equations, one finds that the device current is proportional to the square of a voltage term:

$$I \alpha (V_{GS} - V_T)^2$$

where:

I = device current

V_{GS} = device gate to source voltage

V_T = device threshold voltage.

In the N-channel COPS devices, current is consumed primarily by the load devices. Most of these devices, though not all, are depletion mode devices with the gate and source tied together. Thus, V_{GS} is 0. Therefore, the primary mechanism for current consumption as related to voltage is variation in V_T. The depletion mode load devices in the COPS NMOS microcontrollers have geometries (length is much greater than width) which tend to minimize variations in threshold voltage. There are additional second order effects related to operating voltage, such as effective channel lengths shortening due to increased voltage, which affect current consumption. These effects, however, do not have a major impact on current consumption. Note also that the threshold voltage is affected by process variation. This is one of the areas where the process variation contributes to the device-to-device variation in current consumption. The user can typically expect to see a 5% to 10% variation in current due to operating voltage with the maximum current consumption occurring at maximum operating voltage.

THE EFFECT OF TEMPERATURE

Of the three operating parameters affecting current consumption in the NMOS COPS microcontrollers, temperature has by far the greatest impact. The relationship is given by the following simplified, empirical equation:

$$I(T) = I_O(T/T_O)^{-3/2}$$

where:

T_O = reference junction temperature in °K

T = device junction temperature in °K

I_O = device current at temperature T_O

$I(T)$ = device current at temperature T.

Although this equation is for a single transistor, it can be applied to the entire microcontroller since all the devices are made with the same process and will exhibit the same

characteristics. It should also be noted that the temperatures involved are device junction temperatures. The junction temperature is essentially a function of two items:

$T_j = F(T_A, \theta_{jA})$

where:

T_j = junction temperature

T_A = ambient temperature

θ_{jA} = package thermal characteristic.

The preceding relationship indicates that the package for the device will affect current because the package affects junction temperature. This should not come as a surprise. One need only consider the differences between ceramic and plastic packages to find support for this claim.

For purposes of discussion, it will be assumed that junction temperature is given by the following:

$T_j = T_A + 25°K$

where T_j and T_A are as defined previously. Note that this is an approximation. It is not necessarily true for all packages, or any package. The relationship between junction temperature and ambient temperature is also not necessarily linear. However, the approximation is reasonable and provides a workable framework.

Substituting the junction temperature relationship into the current equation, the following equation results:

$$I(T_A) \cong I_O \left(\frac{T_A + 25}{T_{AO} + 25} \right)^{-3/2}$$

where:

T_{AO} = reference ambient temperature, °K

T_A = ambient temperature, °K

I_O = current at ambient temperature T_{AO}

$I(T_A)$ = current at ambient temperature T_A.

AN EXAMPLE

The COP320L has a specified maximum current of 10 mA. In this process, maximum current occurs at minimum temperature, which is −40°C in this case. It is desired to find the maximum current at 25°C. Therefore,

$T_{AO} = -40°C = 233°K$

$T_A = 25°C = 298°K$

$I_O = 10$ mA

$I(T_A)$ to be determined

$$I(T_A) \cong I_O \left(\frac{T_A + 25}{T_{AO} + 25} \right)^{-3/2}$$

$$\cong 10 \text{ mA } (323/258)$$

$$\cong 7.14 \text{ mA.}$$

Thus the maximum current for the COP320L at 25°C is approximately 7 mA.

CONCLUSION

A means is provided to the user to approximate the current variation of the NMOS COPS microcontroller over its valid operating range. A given device will consume its maximum current at maximum operating voltage, maximum operating frequency, and minimum operating ambient temperature. Conversely, minimum current will be consumed at minimum operating voltage, minimum operating frequency, and maximum operating ambient temperature.

The user should remember that this document is intended as a guide only. The values produced here are reasonable but they are approximations and are not guaranteed values. The user should also remember that the equations and methods discussed here do not involve process variation. The numbers calculated approximate the worst-case maximum current values at a given set of operating conditions. The user should be prepared to see a wide range of values over the course of volume production.

Further Information on Testing of COPS™ Microcontrollers

National Semiconductor
Application Brief 4
Len Distaso

COP Note 7 describes the basic approach and philosophy for testing COPS microcontrollers. This application brief is intended to complement and expand COP Note 7. It is assumed that the reader is familiar with and has access to COP Note 7.

TEST MODE

On COPS microcontrollers, test mode is entered by forcing the SO output to a logic "1" when it should otherwise be a logic "0". The easiest way to do this is to hold the COPS device in reset, hold the $\overline{\text{RESET}}$ pin low, and pull SO up to a logic "1" level. WARNING: *Do not force more than 3.0V on SO, as damage to the device may occur.* SO should be forced to approximately 2.5V to guarantee entry into test mode and to protect the device from damage.

Once the device is in test mode, the state of the SI input controls the type of test. SI at a logic "1" (high level) conditions the device to accept instructions from an external source via the L port. In test mode, when SI is high, the internal ROM is disabled. SI at a logic "0" (low level) forces the device to dump the internal ROM to the L port where the user can read and verify the ROM contents.

INSTRUCTION INPUT

With the device in test mode and SI at a logic "1", the microcontroller will read the data at the L port as instructions. The instructions must be presented at the beginning of each cycle time and must remain valid during the whole cycle time. The chip SK output is the instruction cycle clock in test mode and can be used as the timing reference. *Figure 1* indicates the timing for instruction input using the chip's SK output as the reference. A new instruction must be valid at the L inputs within approximately 200 ns of the rising edge of SK. The user should make every effort to make this time (t2 in *Figure 1*) as short as possible.

It is possible to create an external SK signal which more closely duplicates the internal SK. This requires building a divider from CKI and synchronizing the resultant signal with the device under test. This is significant because it is the internal version of the SK signal which is the master timing signal for the microcontroller. The short time from the rising edge of the SK output to instruction valid is necessary because the actual objective is to provide new instructions at the rising edge, or close to it, of the internal timing signal. If the user creates the external timing signal, the 200 ns time is not applicable. A new instruction, or ROM word, would be presented at each rising edge of the external signal. A method for generating and using this external SK is described in COP Note 7.

ROM DUMP

With SI at logic "0" in test mode, the microcontroller will dump the ROM to the L port. ROM will be dumped sequentially, one word at a time, starting at whatever value the program counter contains. A new ROM word appears at the L lines every falling edge of the chip SK signal. The output timing (t1 in *Figure 1*) is the L output timing as found in the various device data sheets. The device will remain in ROM dump mode as long as SI is at logic "0" in test mode. The program counter will wrap around from the maximum address to 000 and ROM dump will continue.

To get a ROM dump, the user cannot simply enter test mode and force SI to logic "0". Some conditioning of the device is necessary. This requires that the user first go into instruction input mode and set up the device. The suggested sequence is as follows:

1. Enter test mode—pull $\overline{\text{RESET}}$ low, force SO to about 2.5V.
2. Force SI to logic "1" and force 0s on L lines—RESET still low.
3. Force RESET high and input the following sequence to the device:

```
CLRA
JMP    3FC   (modify for ROM size)
LQID
O44H
LEI    4
NOP
```

4. During the NOP, change SI from high to low as shown in *Figure 2*. The ROM dump should start at address 000H at the time shown in *Figure 2*.

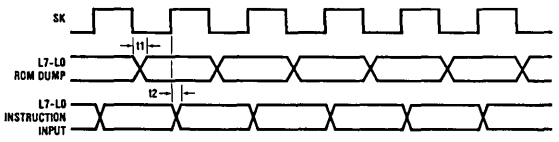
Figure 3 presents a general timing diagram for the entire sequence above. The jump instruction (JMP 3FC) in the sequence is used merely to position the program counter so that the ROM dump will begin at a specified location. That jump will be modified to reflect different ROM sizes or different desired starting locations for the ROM dump.

CHANGING BETWEEN INSTRUCTION INPUT AND ROM DUMP

The change from instruction input to ROM dump is accomplished according to the timing in *Figure 2*. It is necessary to do this to perform a valid ROM dump. However, it is not recommended to go the other direction, from ROM dump to instruction input, "on the fly". The instruction input mode should only be entered while the device is reset, $\overline{\text{RESET}}$ line low, to guarantee proper timing.

CONCLUSION

With COP Note 7 and this application brief, the user should be able to create a workable functional test for his COPS microcontroller. The relative timing is presented here and general techniques and sequences are provided in COP Note 7.

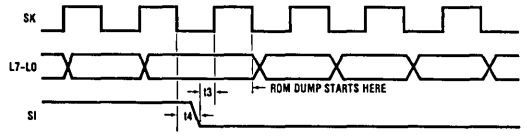


t1 = L output timing (t_{PD1}, t_{PD0}) as found in data sheet

t2 ~ 200 ns max

TL/DD/5146-1

FIGURE 1. Basic Test Mode Timing

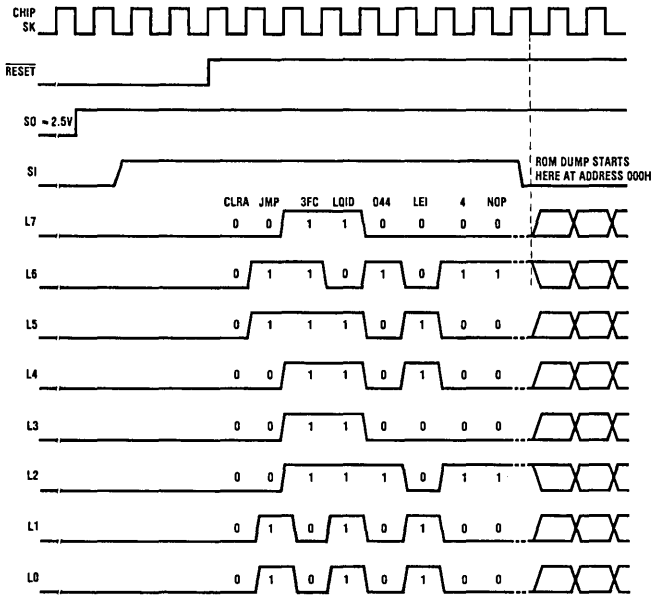


t4 ≈ t3 ~ 1 μs min for 4 μs devices

t4 ≈ t3 ~ 4 μs min for 16 μs devices

TL/DD/5146-2

FIGURE 2. Timing for Changing from Instruction Input to ROM Dump—Test Mode



TL/DD/5146-3

FIGURE 3. Relative Timing for Suggested Sequence to Generate ROM Dump

COPS™ Interrupts

National Semiconductor
Application Brief 6
Jim Murashige

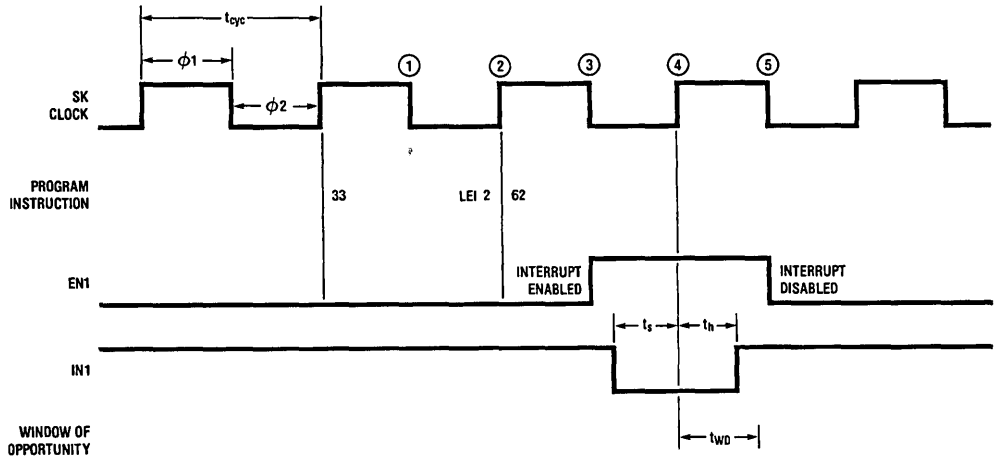
This brief describes in detail the timing requirements pertinent to COPS interrupts. *Figure 1* shows a typical enable-interrupt sequence in relation to the SK (Instruction Cycle) Clock. The SK clock is actually derived afrom the $\phi 1$ clock which is 180° out of phase with the $\phi 2$ clock. It is the $\phi 1$ and $\phi 2$ clocks to which all operation is referenced but for our purposes the SK will suffice. Program instructions are read on a rising $\phi 1$ edge and executed during the $\phi 1$, $\phi 2$ cycle time. Here we see the EN register interrupt enable bit EN2 being set with an LEI instruction. Interrupts are actually enabled on the $\phi 2$ leading edge of the second byte of the instruction point ③. Timing for an INTERRUPT DISABLE is essentially the same.

The interrupt line is sampled on the leading edge of $\phi 1$ as shown and interrupts are recognized if the minimum setup and hold times shown are satisfied. Note that the guaranteed times are longer than the typicals. The interrupt signal conditioning circuitry contains a falling edge detection circuit (a one shot) which requires that in addition to meeting the setup and hold times, the enable interrupt bit EN1 must have been turned on sometime before the end of the WINDOW of OPPORTUNITY shown. If not, the interrupt will be missed and another high to low IN1 transition will be required. EN1 is automatically disabled upon interrupt recognition at point ⑤. Note that although the interrupt is recognized at point ④ it will not be acted upon until all successive transfer of control instructions are executed as defined in the data sheets.

Because of gate delays it is doubtful that if an interrupt had been generated in time to meet the leading $\phi 1$ edge at point ② that the EN1 enable bit would have been on in time to meet the WINDOW of OPPORTUNITY.

By doing a worst case analysis one can see that in order to guarantee reception of an asynchronous interrupt IN1 must remain low for at least 2 instruction cycles. The analysis is as follows. Assuming that interrupts had been enabled prior to point ①, if the interrupt arrives a little after point ① it will not satisfy the minimum setup requirements bringing us up to a point ⑤ our total elapsed time becomes ⑤ − ① = 2 t_{CYC}.

In a dual COPS the interrupt sequence is the same except that now an instruction cycle time is made up of both a Processor X and a Processor Y instruction execution cycle. With one $\phi 1$ and $\phi 2$ clock per processor execution cycle itne instruction cycle time is made up of 2 $\phi 1$'s and $\phi 2$'s. Therefore 1 instruction cycle time in a dual COPS is equivalent to 2 instruction cycle times in a single COPS as far as $\phi 1$'s, $\phi 2$'s and interrupts are concerned.



TL/DD/5180–1

FIGURE 1. COP Interrupt Diagram

Parameter	Min	Typ	Max
t_s	½ t_{CYC}	200 ns	
t_h	½ t_{CYC}	200 ns	
t_{WO}	− ∞	½ t_{CYC} − 600 ns	0

Protecting Data in Serial EEPROMs

National Semiconductor
Application Brief 15
Paul Lubeck

National offers a broad line of serial interface EEPROMs which share a common set of features:

- Low cost
- Single supply in all modes (+5V ± 10%)
- TTL compatible interface
- MICROWIRE™ compatible interface
- Read-Only mode or read-write mode

This Application Brief will address protecting data in any of National's Serial Interface EEPROMs by using read-only mode.

Whereas EEPROM is non-volatile and does not require V_{CC} to retain data, the problem exists that stored data can be destroyed during power transitions. This is due to either uncontrolled interface signals during power transitions or noise on the power supply lines. There are various hardware design considerations which can help eliminate the problem although the simplest most effective method may be the following programming method.

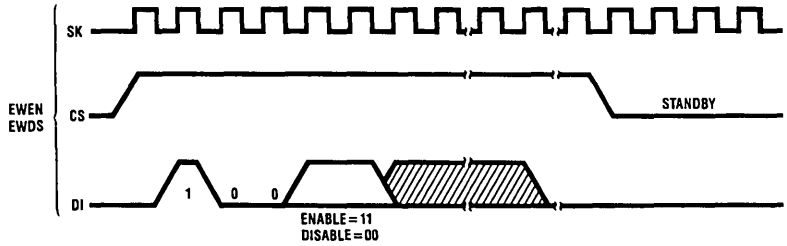
All National Serial EEPROMs, when initially powered up are in the Program Disable Mode*. In this mode it will abort any requested Erase or Write cycles. Prior to Erasing or Writing it is necessary to place the device in the Program Enable Mode†. Following placing the device in the Program Enable Mode, Erase and Write will remain enabled until either executing the Disable instruction or removing V_{CC}. Having V_{CC} unexpectedly removed often results in uncontrolled interface signals which could result in the EEPROM interpreting a programming instruction causing data to be destroyed.

Upon power up the EEPROM will automatically enter the Program Disable Mode. Subsequently the design should incorporate the following to achieve protection of stored data.

1) The device powers up in the read-only mode. However, as a backup, the EWDS instruction should be executed as soon as possible after V_{CC} to the EEPROM is powered up to ensure that it is in the read-only mode.
2) Immediately preceding a programming instruction (ERASE, WRITE, ERAL or WRAL), the EWEN instruction should be executed to enable the device for programming; the EWDS instruction should be executed immediately following the programming instruction to return

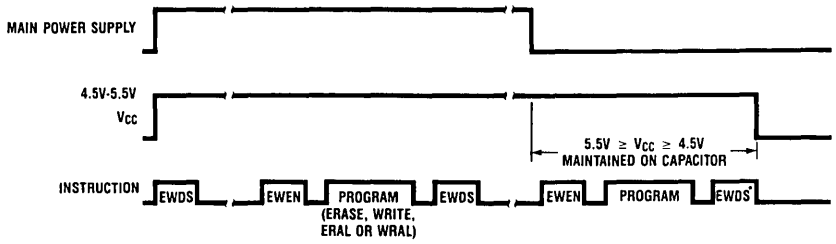
*EWDS or WDS, depending on exact device.

†EWEN or WEN, depending on exact device.



TL/D/7085-1

FIGURE 1. EWEN, EWDS Instruction Timing



TL/D/7085-2

*EWDS must be executed before V_{CC} drops below 4.5V to prevent accidental data loss during subsequent power down and/or power up transients.

FIGURE 2. Typical Instruction Flow for Maximum Data Protection

the device to the read-only mode and protect the stored data from accidental disturb during subsequent power transients or noise.

3) Special care must be taken in designs in which programming instructions are initiated to store data in the EEPROM after the main power supply has gone down. This is usually accomplished by maintaining V_{CC} for the EEPROM and its controller on a capacitor for a sufficient amount of time (approximately 50 ms, depending on the clock rate) to complete these operations. This capacitor must be large enough to maintain V_{CC} between 4.5 and 5.5 volts for the total duration of the store operation, INCLUDING the execution of the EWDS instruction immediately following the last programming instruction. FAILURE TO EXECUTE THE LAST EWDS INSTRUCTION BEFORE V_{CC} DROPS BELOW 4.5 VOLTS MAY CAUSE INADVERTENT DATA DISTURB DURING SUBSEQUENT POWER DOWN AND/OR POWER UP TRANSIENTS.

COPS™ Peripheral Chips

National Semiconductor
Application Brief 28

There are several I/O peripheral chips that are compatible with the COPS microcontroller by communicating through the serial I/O port.

Two different sets of timing employed by them are shown in *Figure 2*. A brief description of the electrical characteristics of each chip is given below.

COP452 FREQUENCY/COUNTER PERIPHERAL

The COP452 is fabricated using N-channel silicon-gate MOS technology. Containing 2 independent 16-bit counter/register pairs, it is well suited to a wide variety of tasks involving the measurement and/or generation of times and/or frequencies. Included are multiple tones, precise duty cycles, event counting, waveform measurement, "white noise" generation, and A-D/D-A conversions. An on-chip zero-crossing detector can trigger a pulse with a programmed delay and duration.

COP470 V.F. DISPLAY DRIVER

The COP470 is designed to directly drive a multiplexed Vacuum Fluorescent display. Data is loaded serially and held in internal latches. The COP470 has an on-chip oscillator to multiplex four digits of eight segment display, and may be cascaded and/or stacked to drive more digits, more segments, or both.

With the addition of external drivers, the COP470 also provides a convenient means of interfacing to a large-digit LED display.

COP472 LIQUID CRYSTAL DISPLAY CONTROLLER

The COP472 Liquid Crystal Display (LCD) Controller is fabricated using CMOS technology. It drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472 contains an on-chip oscillator and generates all the multilevel waveforms for backplanes and segment outputs on a triplex display. One COP472 can drive 36 segment multiplexed as 3 x 12 (4½ digit display). Two COP472 devices can be used together to drive 72 segments (3 x 24) which could be an 8½ digit display.

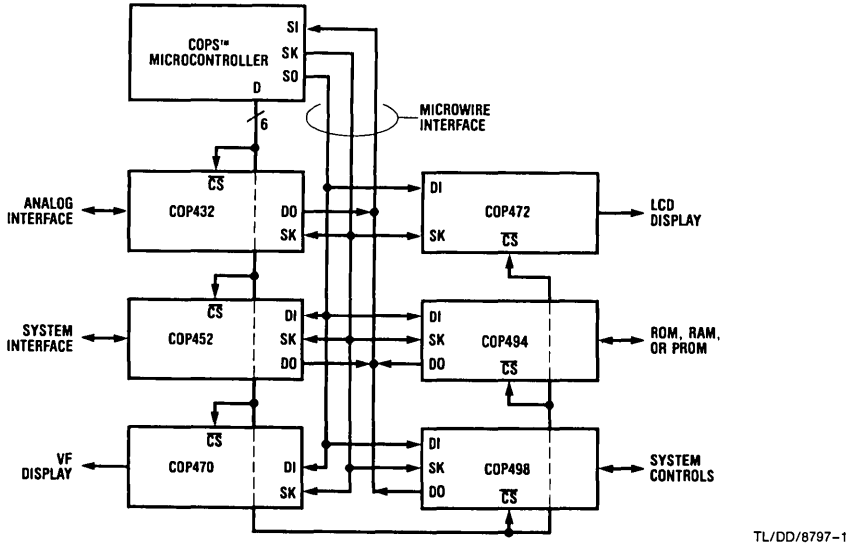
COP494 256-Bit Serial Electrically erasable programmable memory. The COP494 is a 256-bit non-volatile memory. The device contains 256 bits of Read/Write memory divided into 16 registers of 16 bits each. Each register is serially read or written by the COP400 controller. Written information is stored in a floating gate cell with at least 10 years retention.

COP498/COP499 LOW POWER CMOS RAM AND TIMER

The COP498 low power CMOS Random-Access Memory and Timer is an external memory and timer chip with the simple MICROWIRE™ serial interface. The device contains 256 bits of read/write memory divided into 4 registers of 64 bits each. The COP498 also contains a crystal based timer for timekeeping purposes, and can provide a "wake-up" signal to turn on a COPS controller.

The COP498 can be used for low power standby memory and can also be used for low power operation by turning the controller off and on, on a duty cycle basis.

The COP499 Low Power CMOS Random-Access Memory is an external memory and switch chip with the simple MICROWIRE serial interface. The device contains 256 bits of read/write memory divided into 4 registers of 64 bits each. The COP499 also contains circuitry that enables the user to turn a controller on and off while maintaining the integrity of the memory.



TL/DD/8797-1

FIGURE 1

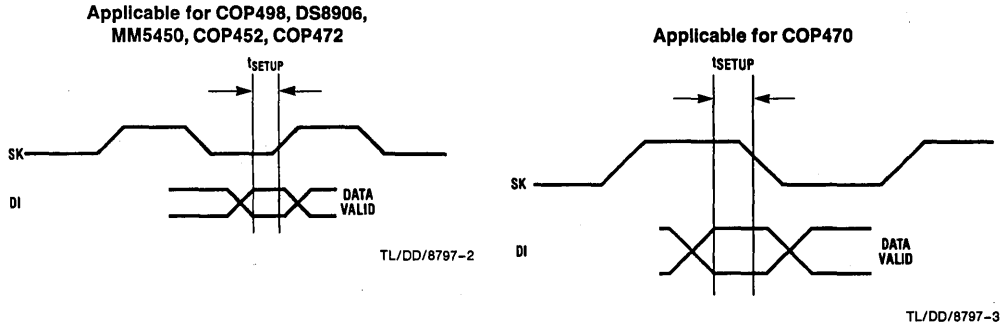


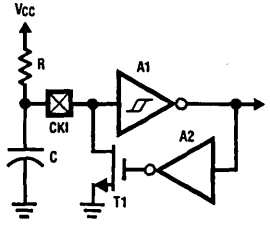
FIGURE 2. Serial Input Data Timing

A Users Guide to COPS™ Oscillator Operation

National Semiconductor
Application Note 326
Jim Murashige

The following discussion is an overview of the COPS oscillator circuits meant to give the reader a working knowledge of the circuits. Although the descriptions are very general and light on detail; a background in complex frequency analysis is necessary. For additional information the references cited should be consulted as well as the many works on oscillator theory.

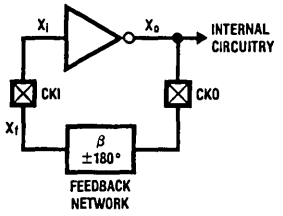
There are 2 basic circuits from which all of the COPS oscillator options are provided. (See option lists in individual data sheets.) The first and simplest in description is the astable one shot of *Figure 1* which gives us our RC oscillator option. A1 and A2 are inverters with A1 possessing a Schmitt trigger input. T1 is a large N channel enhancement MOS FET. Operation with the external R-C shown is as follows. Assuming C is initially discharged the CKI pin is low forcing T1 off. As C charges through R the trigger point of A1 is eventually reached at which time T1 is turned on discharging C and beginning a new cycle. Although almost any combination of R-C could be chosen, we would ideally like to have as short a discharge time as possible thereby eliminating the high variability in T1 drain current from device to device as a timing factor. For this reason R is chosen very large and C very small. This choice also leads to minimum R-C power dissipation. For the CKI Schmitt trigger clock input option the T1 MOS FET is merely mask disabled from the oscillator circuit.



TL/DD/5139-1

FIGURE 1. R-C Oscillator

The second oscillator circuit is the classic phase shift oscillator depicted in *Figure 2*. Found not only on COPS but on most other microprocessor circuits it is the simplest oscillator in terms of component complexity but the most difficult to analyze.



TL/DD/5139-2

FIGURE 2. Phase Shift Oscillator

The conditions under which the circuit will oscillate are described by the Barkhausen Criterion which states that oscillation will occur at the frequency for which the total loop phase shift from x_i to x_f is 0° or a multiple of 360° (i. e., x_f is identical to x_i). In addition the total loop gain must be > 1 to insure self propagation. The inverting amplifier shown between x_i and x_o provides 180° of phase shift thus leaving the feedback network to supply the other ±180°. The feedback network can be comprised of active or passive components but highly effective oscillators are possible using only passive reactive components and the general configuration of *Figure 3*.

If you work out the feedback loop equations for *Figure 3* it can be shown that in order to achieve ±180° phase shift:

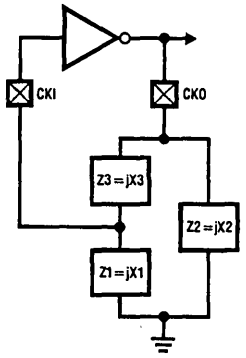
$$X1 + X2 + X3 = 0 \qquad (1)$$

X1 and X2 must both be inductors or capacitors (2)

therefore X3 is inductive if X1 is capacitive and vice versa

if X1 and X2 are capacitors it is a Colpitts Oscillator

X1 and X2 are inductors it is a Hartley Oscillator



TL/DD/5139-3

FIGURE 3. Typical Feedback Configuration

The Colpitts configuration is commonly shown in microprocessor oscillator circuits (*Figure 5*) with the inductive X3 replaced by a crystal for reasons we shall soon see. The equivalent electrical model of a crystal is shown in *Figure 4b* and a plot of its Reactance versus Frequency shown in *Figure 4c*. R-L-C represent the electro-mechanical properties of the crystal and C_0 the electrode capacitance. There are 2 important points on the reactance curve labeled f_a and f_b.

$$\text{At } f_a = \frac{1}{2\pi}\sqrt{\frac{1}{LC}}$$

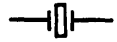
the crystal is at series resonance with L and C canceling each other out leaving only a nonreactive R for 0 phase shift. This mode of operation is important in oscillator circuits where a non-inverting amplifier is used and 0° phase shift must be preserved.

$$\text{At } f_b = \frac{1}{2\pi}\sqrt{\frac{1}{LC} + \frac{1}{LC_C}}$$

which is just a little higher than f_a the crystal is at parallel resonance and appears very inductive or capacitive. Note that the cyrstal will only appear inductive between f_a and f_b and that it becomes highly inductive very quickly. In addition f_b is only a fraction of a percent higher than f_a. Therefore the only time that the crystal will satisfy the X3 = −(X1 + X2) condition in the Colpitts configuration of *Figure 5* is when the circuit is oscillating between f_a and f_b. The exact frequency will be the one which gives an inductive reactance large enough to cancel out:

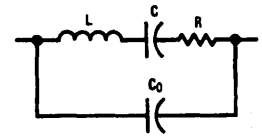
$$X1 + X2 = \frac{1}{\omega C1} + \frac{1}{\omega C2} = \frac{1}{\omega}\left[\frac{1}{C1} + \frac{1}{C2}\right] = \frac{1}{2\pi f}\left[\frac{1}{C_L}\right]$$

Therefore by varying C1 or C2 we can trim slightly the oscillator frequency.



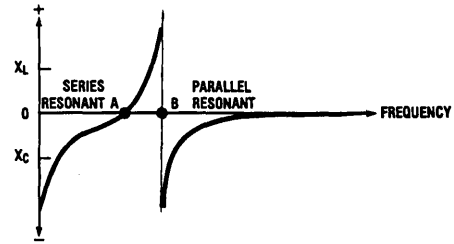
TL/DD/5139-4

a. Circuit Symbol



TL/DD/5139-5

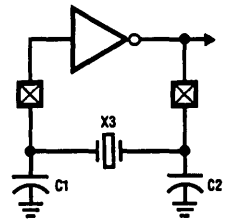
b. Electrical Equivalent



TL/DD/5139-6

c. Reactance Versus Frequency

FIGURE 4. Quartz Crystal

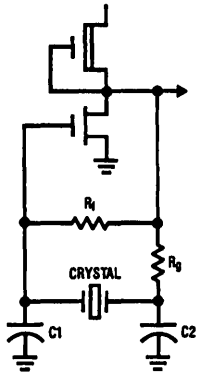


TL/DD/5139-7

FIGURE 5. Colpitts Oscillator

The Q of a circuit is often bounced around in comparing different circuits and can be viewed graphically here as the slope of the reactance curve between f_a and f_b. Obviously the steeper the curve the smaller the variation in f necessary to restore the Barkhausen Phase Shift Criterion. In addition a lower Q (more R) means that the reactance curve won't peak as high at f_b, necessitating a smaller X1 + X2. When selecting crystals the user should be aware that the frequency stamped on the cans are for either parallel or series resonance, which, although very close, may matter significantly in the particular application.

An actual MOS circuit implementation of *Figure 5* is shown in *Figure 6*. It consists of a MOS inverter with depletion load and the crystal π network just presented. External to the COPS chips are the R_f and R_g resistors. R_f provides bias to the MOS inverter gate $V_g = V_o$. Since the gate draws no current R_f can be very large (MΩ) and should be, since we do not wish it to interact with the crystal network. R_g increases the output resistance of the inverter and keeps the crystal from being over driven.

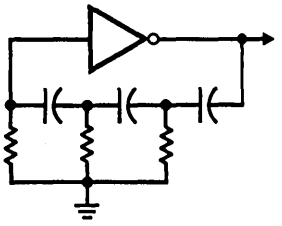


TL/DD/5139-8

FIGURE 6. MOS Oscillator

Of course the feedback network doesn't have to have the configuration of *Figure 3* and can be anything so long as the Barkhausen Phase Shift Criterion is satisfied. One popular configuration is shown in *Figure 7* where the phase shift will be 180°

$$\text{at } f = \frac{1}{(2\pi RC\sqrt{6})}$$



TL/DD/5139-9

FIGURE 7. R-C Phase Shift Oscillator

REFERENCES

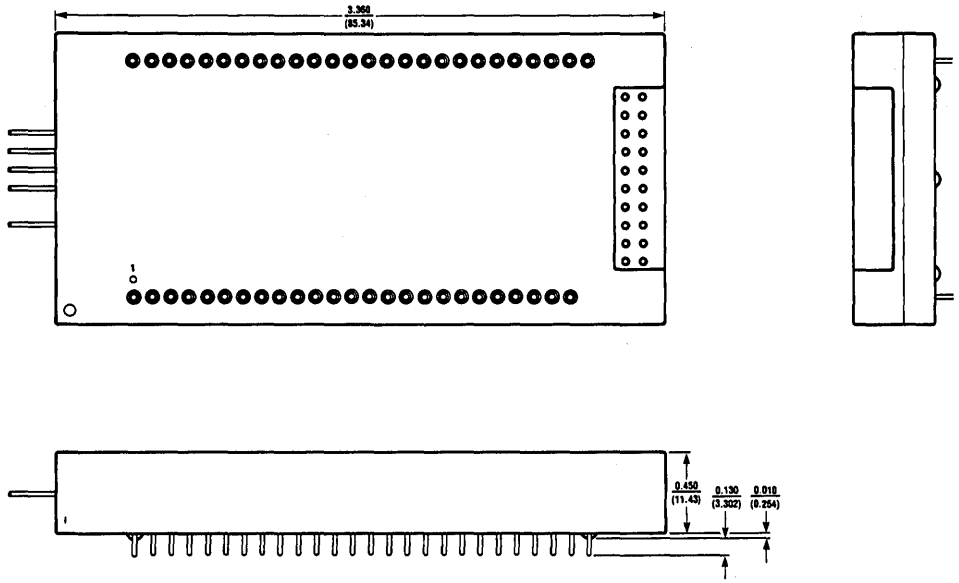
1. Crystal/INS8048 Oscillator, AN-296, March 1982, National Semiconductor

2. Oscillator Characteristics of COPS Microcontrollers, CN-5, Feb. 1981, National Semiconductor

3. Integrated Electronics, Chapter 14, Millman and Halkias 1972

4. Handbook of Electronics Calculations, Chapter 9, Kaufman and Seidman 1979

5. 1982 COPS Microcontroller Databook, National Semiconductor



TL/DD/5139-10

Implementing an 8-Bit Buffer in COPS™

National Semiconductor
Application Note 329
David Pointer

Sometimes a COP microcontroller must input and/or output 8-bit data; for instance, when handling ASCII data. In some applications, the processor must also provide temporary storage for 8-bit data before it is output. The COP instruction set and RAM structure lend themselves very nicely to providing a 32 digit, 8-bit buffer for a solution to these applications.

Such a large buffer is possible using a COP440 or a COP444L. The other members of the COP400 family with half as much RAM as these two would provide a 16 digit 8-bit buffer using the techniques described in this example.

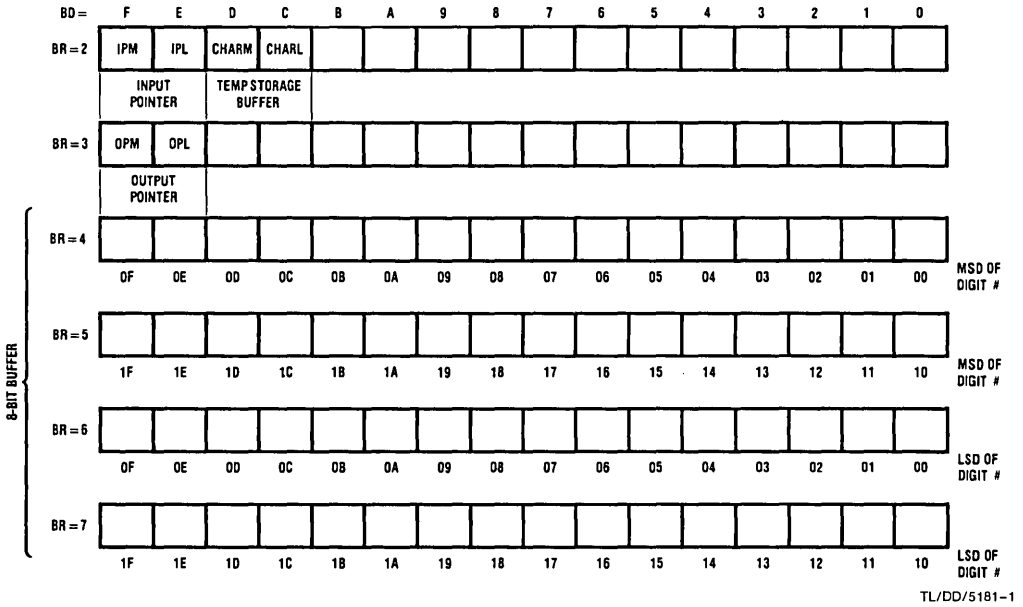
Four adjacent RAM registers (16 digits each) are required. Referring to *Figure 1*, registers 4, 5, 6, and 7 are used for the buffer. Each RAM location contains 4 bits, so 2 locations will be used to store a byte of data. But these RAM locations are not adjacent to each other. You will note that the MSD of digit number 0A hex is in RAM location (4, A) while the LSD of the same digit is in RAM location (6, A).

The 2 RAM locations CHARM and CHARL are used for temporary storage of an 8-bit value.

In addition, 4 RAM locations are used for buffer pointers: those labelled IPM and IPL are the MSD and LSD of the input pointer, and those labelled OPM and OPL are the MSD and LSD of the output pointer. Each pointer's function is to store an 8-bit counter whose value ranges from 00 hex thru 1F hex. The input pointer's value is used for storing the temporary storage buffer contents into the digit with the same number. For example, if the input pointer equals 14 hex, then the contents of CHARM would be stored in RAM location (5, 4) and the contents of CHARL would be stored in RAM location (7, 4). The output pointer's value is used for retrieving a digit from the buffer and putting it in CHARM and CHARL. For instance, if the output pointer equals 05 hex, then the contents of RAM location (4, 5) would be transferred to CHARM and the contents of RAM location (6, 5) would be transferred to CHARL.

A simple example of one possible application of the buffer is flowcharted in *Figure 2.* In this example, data is input to CHARM and CHARL, then stored in the buffer. An output device (a printer) is checked to see if it is ready to receive data. If it is, data is brought out of the buffer and put in CHARM and CHARL for output to the printer.

Pages 3 and 4 contain a listing of the subroutines needed to perform the data transfers in the 32-digit, 8-bit buffer.



TL/DD/5181-1

FIGURE 1. 8-Bit Buffer RAM Map

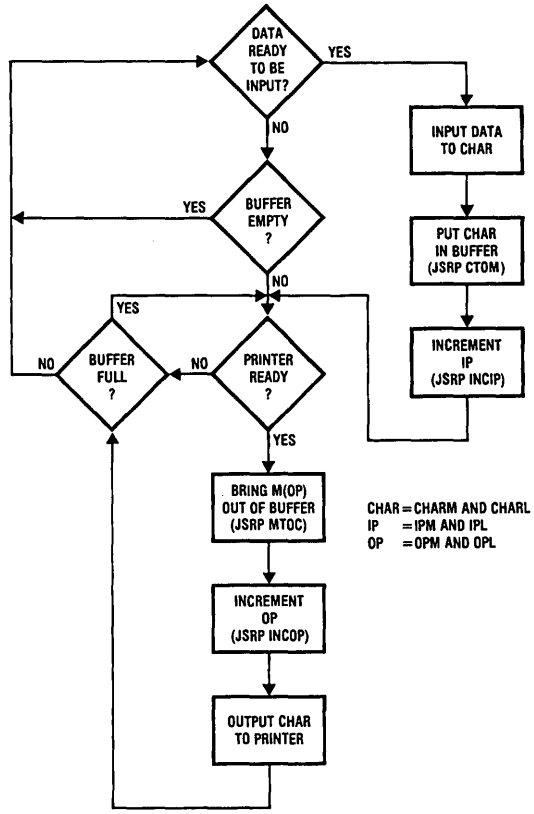


FIGURE 2. Buffer Example Flowchart

```
COP CROSS ASSEMBLER    PAGE:  1
BUFFER

    1            ;************************************
    2            ;***                              ***
    3            ;***  8-BIT RAM BUFFER SUBROUTINES  ***
    4            ;***                              ***
    5            ;************************************
    6            ;THESE ARE SUBROUTINES FOR IMPLEMENTING A 32 BYTE
    7            ;BUFFER IN A COP440 OR COP444L RAM  9/3/82
    8     01BC   .CHIP 444
    9            .TITLE BUFFER
   10     002D   CHARM   =       2,13          ;TEMPORARY STORAGE BUFFER MSD
   11     002C   CHARL   =       2,12          ;TEMPORARY STORAGE BUFFER LSD
   12     002F   IPM     =       2,15          ;INPUT POINTER MSD
   13     002E   IPL     =       2,14          ;INPUT POINTER LSD
   14     003F   OPM     =       3,15          ;OUTPUT POINTER MSD
   15     003E   OPL     =       3,14          ;OUTPUT POINTER LSD
   16 000 00             CLRA
   17     0080   .PAGE 2
   18            ;MTOC IS A SUBROUTINE THAT TRANSFERS M(OPM) AND M(OPL) TO
   19            ;CHARM AND CHARL
   20 080 233E   MTOC:    LDD     OPL           ;LOAD LSD OUTPUT POINTER
   21 082 50              CAB                   ;WHICH IS BD
   22 083 233F            LDD     OPM            ;LOAD MSB OUTPUT POINTER FOR B
   23 085 54              AISC    4              ;MAKE BR EQUAL 4 OR 5
   24 086 12              XABR
   25 087 25              LD      2             ;LOAD M(OPM), MAKE BR = 6 OR 7
   26 088 23AD            XAD     CHARM         ;M(OPM) TO CHARM
   27 08A 05              LD                     ;LOAD M(OPL)
   28 08B 23AC            XAD     CHARL         ;M(OPL) TO CHARL
   29 08D 48              RET
   30            ;
   31            ;
   32            ;CTOM IS A SUBROUTINE THAT TRANSFERS CHARM AND CHARL TO
   33            ;M(IPM) AND M(IPL)
   34 08E 232E   CTOM:    LDD     IPL           ;LOAD LSD INPUT POINTER
   35 090 50              CAB                   ;WHICH IS BD
   36 091 232F            LDD     IPM           ;LOAD MSD INPUT POINTER FOR BR
   37 093 54              AISC    4             ;MAKE BR = 4 OR 5
   38 094 12              XABR
   39 095 232D            LDD     CHARM         ;LOAD MSD TEMP STORAGE
   40 097 26              X       2             ;TO M(OPM), MAKE BR = 6 OR 7
   41 098 232C            LDD     CHARL         ;LOAD LSD TEMP STORAGE
   42 09A 06              X                     ;TO M(OPL)
   43 09B 48              RET
   44            ;
   45            ;
```

```
COP CROSS ASSEMBLER    PAGE:  2
BUFFER

   46              .FORM
   47              ;INCREMENTS INPUT POINT OR OUTPUT POINTER, ROLLS OVER
   48              ;AT 1F HEX
   49 09C 2D     INCIP:  LBI     IPL             ;POINT TO LSD OF POINTER
   50 09D 3D     INCOP:  LBI     OPL
   51 09E 22             SC                      ;C=1 FOR INCREMENT
   52 09F 00             CLRA
   53 0A0 30             ASC                     ;INCREMENT RAM VALUE
   54 0A1 44             NOP                     ;NEGATES SKIP CONDITION
   55 0A2 04             XIS                     ;STORE AND POINT TO (X,F)
   56 0A3 00             CLRA
   57 0A4 30             ASC                     ;PROPAGATE CARRY, IF ANY, TO MS
   58 0A5 44             NOP
   59 0A6 06             X                       ;STORE
   60 0A7 45             RMB     1               ;ROLL OVER AT X'1F
   61 0A8 48             RET
   62              ;
   63              ;
   64              .END


COP CROSS ASSEMBLER    PAGE:  3
BUFFER

CHARL  002C     CHARM  002D     CTOM   008E *   INCIP  009C *
INCOP  009D *   IPL    002E     IPM    002F     MTOC   0080 *
OPL    003E     OPM    003F
NO ERROR LINES
   42  ROM WORDS USED
COP   444  ASSEMBLY
SOURCE CHECKSUM = C6A5
INPUT FILE    6:RBUFFC. SRC  VN:    5
```

Designing with the NMC9306/COP494 a Versatile Simple to Use E^2 PROM

National Semiconductor
Application Note 338
Masood Alavi

This application note outlines various methods of interfacing an NMC9306/COP494 with the COPS™ family of microcontrollers and other microprocessors. *Figures 1–6* show pin connections involved in such interfaces. *Figure 7* shows how parallel data can be converted into a serial format to be inputted to the NMC9306; as well as how serial data outputted from an NMC9306 can be converted to a parallel-format.

The second part of the application note summarizes the key points covering the critical electrical specifications to be kept in mind when using the NMC9306/COP494.

The third part of the application note shows a list of various applications that can use a NMC9306/COP494.

GENERIC CONSIDERATIONS

A typical application should meet the following generic criteria:

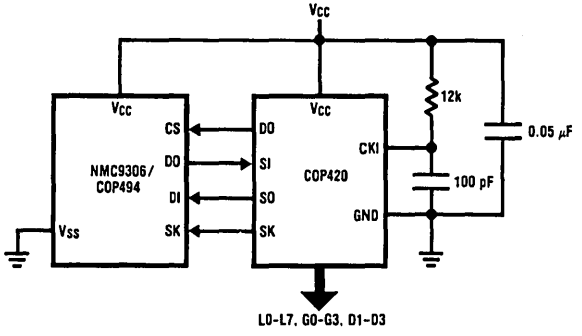
1. Allow for no more than 10,000 E/W cycles for optimum and reliable performance.
2. Allow for any number of read cycles.
3. Allow for an erase or write cycle that operates in the 10–30 ms range, and not in the tens or hundreds of ns range as used in writing RAMs. (Read vs write speeds are distinctly different by orders of magnitude in E^2PROM, not so in RAMs.)
4. No battery back-up required for data-retention, which is fully non-volatile for at least 10 years at room-ambient.

SYSTEM CONSIDERATIONS

When the control processor is turned on and off, power supply transitions between ground and operating voltage may cause undesired pulses to occur on data, address and control lines. By using WEEN and WEDS instructions in conjunction with a LO-HI transition on CS, accidental erasing or writing into the memory is prevented.

The duty cycle in conjunction with the maximum frequency translates into having a minimum Hi-time on the SK clock. If the minimum SK clock high time is greater than 1 μs, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max. On the low side no limit exists on the minimum frequency. This makes it superior to the COP499 CMOS-RAM. The rise and fall times on the SK clock can also be slow enough not to require termination up to reasonable cable-lengths.

Since the device operates off of a simple 5V supply, the signal levels on the inputs are non-critical and may be operated anywhere within the specified input range.



TL/D/5286–1

FIGURE 1. NMC9306/COP494 — COP420 Interface

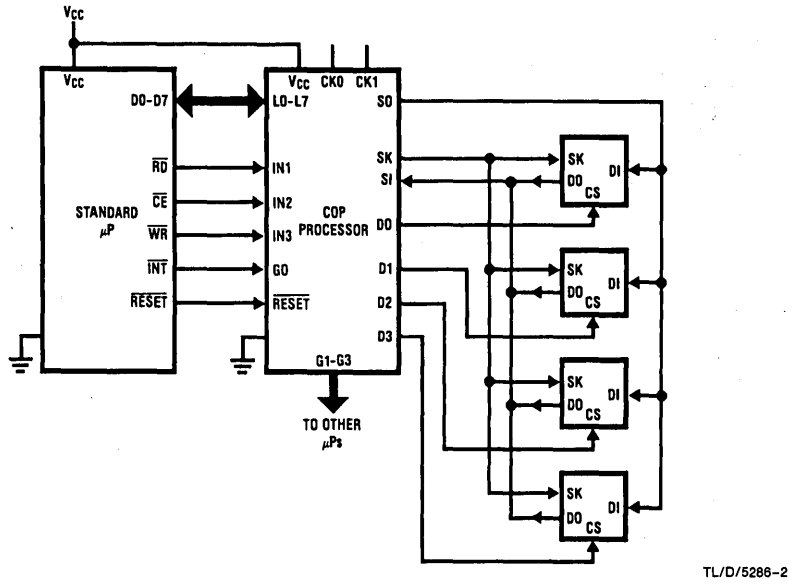
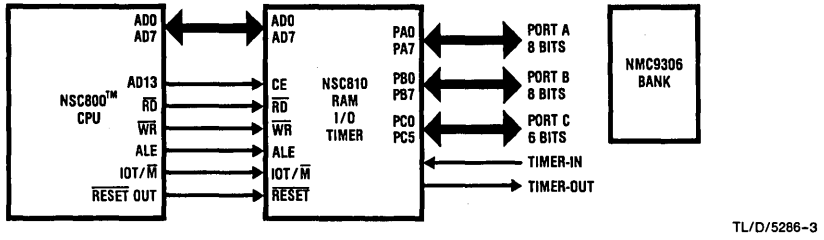


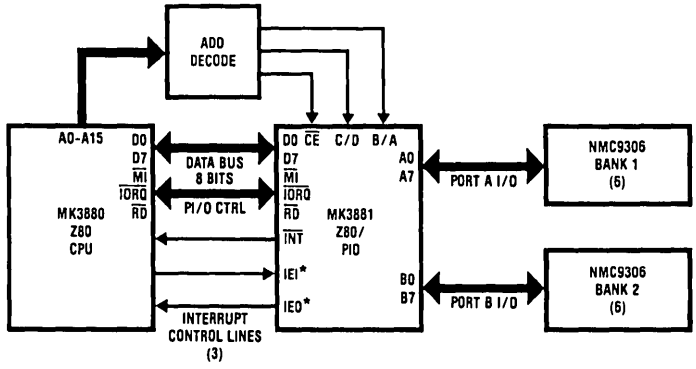
FIGURE 2. NMC93O6 — Standard μP Interface Via COP Processor



PA0 → SK
PA1 → DI/DO } Common to all 9306's
PA2–7 → 6CS for 6– 9306's

* SK is generated on port pins by bit-set and bit-clear operations in software. A symmetrical duty cycle is not critical.
* CS is set in software. To generate 10–30 ms write/erase the timer/counter is used. During write/erase. SK may be turned off.

FIGURE 3. NSC800™ to NMC9306 Interface (also Valid for 8085/8085A and 8156)

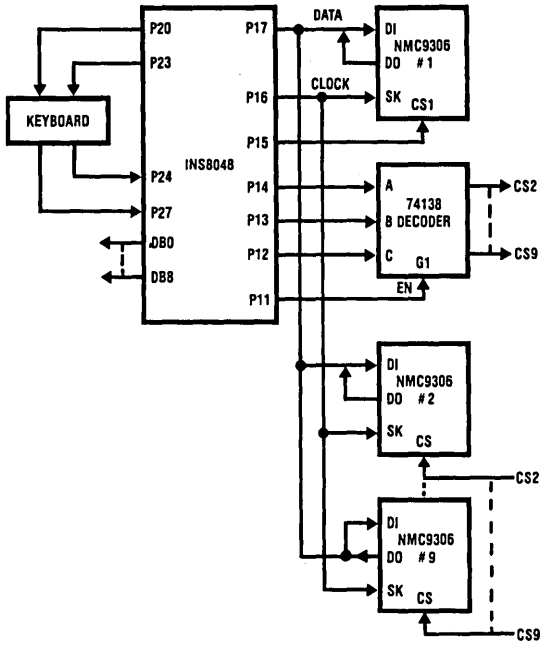


TL/D/5286–4

Z80-P10	9306	
A0	SK	Common to all 9306's (Bank 1)
A1	DI/DO	
A2-A7	CS1-CS6	

* Only used if priority interrupt daisy chain is desired
* Identical connection for Port B

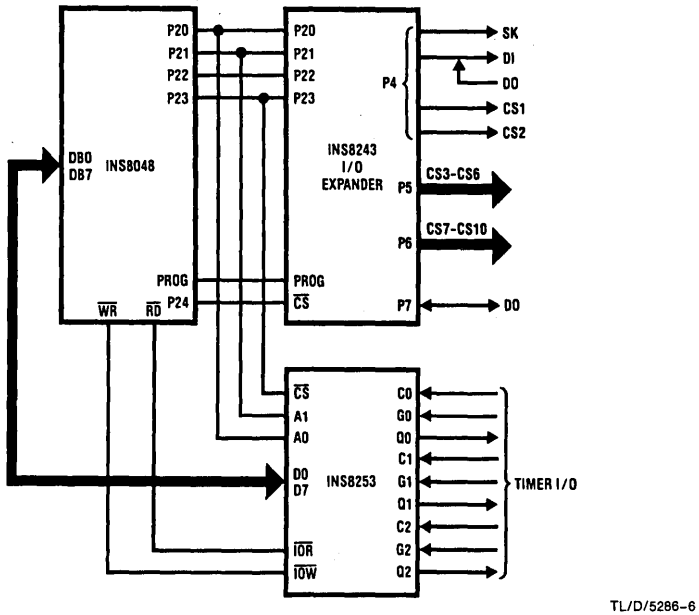
FIGURE 4. Z80 — NMC9306 Interface Using Z80-PIO Chip



TL/D/5286–5

* SK and DI are generated by software. It should be noted that at 2.72 μs/instruction. The minimum SK period achievable will be 10.88 μs or 92 kHz, well within the NMC9306 frequency range.
* DO may be brought out on a separate port pin if desired.

FIGURE 5. 48 Series μP — NMC9306 Interface

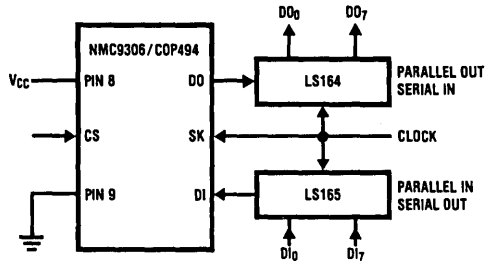


TL/D/5286-6

Expander outputs

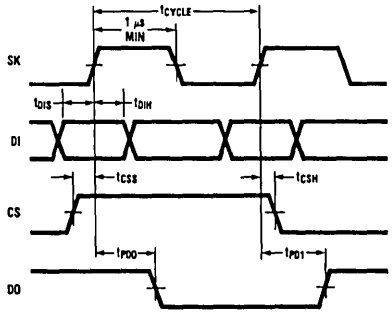
	DI SK	(COMMON)
Port 4	CS1	
	CS2	
Port 5-6	CS3–CS10	
Port 7	DO (COMMON)	

FIGURE 6. 8048 I/O Expansion



TL/D/5286-7

FIGURE 7. Converting Parallel Data Into Serial Input for NMC9306/COP494



Min	Max
t_{CYCLE} 0	250 kHz
t_{DIS} 400	ns
t_{D1H} 400	ns
t_{CSS} 200	ns
t_{CSH} 0	ns
t_{PD0}	2 μs
t_{PD1}	2 μs

TL/D/5286–8

FIGURE 8. NMC9306/COP494 Timing

THE NMC9306/COP494

Extremely simple to interface with any μP or hardware logic. The device has six pins for the following functions:

Pin 1	CS*	HI enabled
Pin 2	SK	Serial Clock input
Pin 3	DI	For instruction or data input
Pin 4	DO**	For data read, TRI-STATE® otherwise
Pin 5	GND	
Pin 8	V_{CC}	For 5V power
Pins 6–7	No Connect	No termination required

*Following an E/W instruction feed, CS is also toggled low for 10 ms (typical) for an E/W operation. This internally turns the VPP generator on (HI-LO on CS) and off (LO-HI on CS).

**DI and DO can be on a common line since DO is TRI-STATED when unselected DO is only on in the read mode.

USING THE NMC9306/COP494

The following points are worth noting:

1. SK clock frequency should be in the 0–250 kHz range. With most μPs this is easily achieved when implemented in software by bit-set and bit-clear instructions, which take 4 instructions to execute a clock or a frequency in the 100 kHz range for standard μP speeds. Symmetrical duty cycle is irrelevant if SK HI time is ≥ 2 μs.
2. CS low period following an E/W instruction must not exceed the 30 ms max. It should best be set at typical or minimum spec of 10 ms. This is easily done by timer or a software connect. The reason is that it minimizes the 'on time' for the high V_{PP} internal voltage, and so maximizes endurance. SK-clock during this period may be turned off if desired.
3. All E/W instructions must be preceded by EWEN and should be followed by an EWDS. This is to secure the stored data and avoid inadvertent erase or write.
4. A continuously 'on' SK clock does not hurt the stored data. Proper sequencing of instructions and data on DI is essential to proper operation.
5. Stored data is fully non-volatile for a minimum of ten years independent of V_{CC}, which may be on or off. Read cycles have no adverse effects on data retention.
6. Up to 10,000 E/W cycles/register are possible. Under typical conditions, this number may actually approach 1 million. For applications requiring a large number of cycles, redundant use of internal registers beyond 10,000 cycles is recommended.
7. Data shows a fairly constant E/W Programming behavior over temperature. In this sense E^2PROMs supersede EPROMs which are restricted to room temperature programming.
8. As shown in the timing diagrams, the start bit on DI must be set by a ZERO - ONE transition following a CS enable (ZERO - ONE), when executing any instruction. ONE CS enable transition can only execute ONE instruction.
9. In the read mode, following an instruction and data train, the DI can be a don't care, while the data is being outputted i.e., for next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
10. The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI till another CS LO-HI transition starts a new instruction cycle.
11. When a common line is used for DI and DO, a probable overlap occurs between the last bit on DI and start bit on DO.
12. After a read cycle, the CS must be brought low for 1 SK clock cycle before another instruction cycle can start.

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ — After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE — Write command shifted in followed by data in (16 bits) then CS pulsed low for 10 ms minimum.

3

INSTRUCTION SET

Instruction	SB	Opcode	Address	Data	Comments
READ	01	10xx	A3A2A1A0		Read Register A3A2A1A0
WRITE	01	01xx	A3A2A1A0	D15–D0	Write Register A3A2A1A0
ERASE	01	11xx	A3A2A1A0		Erase Register A3A2A1A0
EWEN	01	0011	XXXX		Erase/Write Enable
EWDS	01	0000	XXXX		Erase/Write Disable
ERAL	01	0010	XXXX		Erase All Registers
WRAL	01	0001	XXXX	D15–D0	Write All Registers

NMC9306 has 7 instructions as shown. Note that MSB of any given instruction is a "1" and is viewed as a start bit in the interface sequence. The next 8 bits carry the op code and the 4-bit address for 1 of 16, 16-bit registers.

X is a don't care state.

The following is a list of various systems that could use a NMC9306/COP494

A. Airline terminal
Alarm system
Analog switch network
Auto calibration system
Automobile odometer
Auto engine control
Avionics fire control

B. Bathroom scale
Blood analyzer
Bus interface

C. Cable T.V. tuner
CAD graphics
Calibration device
Calculator—user programmable
Camera system
Code identifier
Communications controller
Computer terminal
Control panel
Crystal oscillator

D. Data acquisition system
Data terminal

E. Electronic circuit breaker
Electronic DIP switch
Electronic potentiometer
Emissions analyzer
Encryption system
Energy management system

F. Flow computer
Frequency synthesizer
Fuel computer

G. Gas analyzer
Gasoline pump

H. Home energy management
Hotel lock

I. Industrial control
Instrumentation

J. Joulemeter

K. Keyboard -softkey

L. Laser machine tool

M. Machine control
Machine process control
Medical imaging
Memory bank selection
Message center control
Mobile telephone
Modem
Motion picture projector

N. Navigation receiver
Network system
Number comparison

O. Oilfield equipment

P. PABX
Patient monitoring
Plasma display driver
Postal scale
Process control
Programmable communications
Protocol converter

Q. Quiescent current meter

R. Radio tuner
Radar dectector
Refinery controller
Repeater
Repertory dialer

S. Secure communications system
Self diagnostic test equipment
Sona-Bouy
Spectral scanner
Spectrum analyzer

T. Telecommunications switching system
Teleconferencing system
Telephone dialing system
T.V. tuner
Terminal
Test equipment
Test system
TouchTone dialers
Traffic signal controller

U. Ultrasound diagnostics
Utility telemetering

V. Video games
Video tape system
Voice/data phone switch

W. Winchester disk controller

X. X-ray machine
Xenon lamp system

Y. YAG—laser controller

Z. Zone/perimeter alarm system

A Study of the Crystal Oscillator for CMOS-COPS™

National Semiconductor
Application Note 400
Abdul Aleaf

INTRODUCTION

The most important characteristic of CMOS-COPS is its low power consumption. This low power feature does not exist in TTL and NMOS systems which require the selection of low power IC's and external components to reduce power consumption.

The optimization of external components helps decrease the power consumption of CMOS-COPS based systems even more.

A major contributor to power consumption is the crystal oscillator circuitry.

Table I presents experimentally observed data which compares the current drain of a crystal oscillator vs. an external squarewave clock source.

The main purpose of this application note is to provide experimentally observed phenomena and discuss the selection of suitable oscillator circuits that cover the frequency range of the CMOS-COPS.

Table I clearly shows that an unoptimized crystal oscillator draws more current than an external squarewave clock. An RC oscillator draws even more current because of the slow rising signal at the CKI input.

Although there are few components involved in the design of the oscillator, several effects must be considered. If the requirement is only for a circuit at a standard frequency which starts up reliably regardless of precise frequency stability, power dissipation and etc., then the user could directly consult the data book and select a suitable circuit with proper components. If power consumption is a major requirement, then reading this application note might be helpful.

WHICH IS THE BEST OSCILLATOR CIRCUIT?

The Pierce Oscillator has many desirable characteristics. It provides a large output signal and drives the crystal at a low power level. The low power level leads to low power dissipation, especially at higher frequencies. The circuit has good short-term stability, good waveforms at the crystal, a frequency which is independent of power supply and temperature changes, low cost and usable at any frequency. As compared with other oscillator circuits, this circuit is not disturbed very much by connecting a scope probe at any point in the circuit, because it is a stable circuit and has low impedance. This makes it easier to monitor the circuit without any major disturbance. The Pierce oscillator has one disadvantage. The amplifier used in the circuit must have high gain to compensate for high gain losses in the circuitry surrounding the crystal.

TABLE I

A. Crystal oscillator vs. external squarewave COP410C change in current consumption as a function of frequency and voltage, chip held in reset, CKI is ÷4.

I = total power supply current drain (at V_{CC}).

Crystal

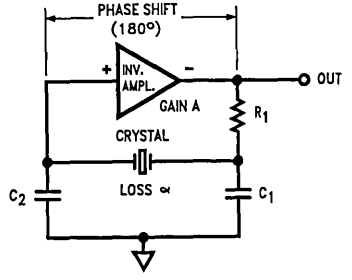
V_{CC}	f_{CKI}	Inst. cyc. time	I μA
2.4V	32 kHz	125 μs	8.5
5.0V	32 kHz	125 μs	83
2.4V	1 MHz	4 μs	199
5.0V	1 MHz	4 μs	360

External Squarewave

V_{CC}	f_{CKI}	Inst. cyc. time	I
2.4V	32 kHz	125 μs	4.4 μA
5.0V	32 kHz	125 μs	10 μA
2.4V	1 MHz	4 μs	127 μA
5.0V	1 MHz	4 μs	283 μA

WHAT IS A PIERCE OSCILLATOR?

The Pierce is a series resonant circuit, and its basic configuration is shown below.



TL/DD/8439–1

FIGURE 1

For oscillation to occur, the Barkhausen criteria must be met: (1) The loop gain must be greater than one. (2) The phase shift around the loop must be 360°.

Ideally, the inverting amplifier provides 180°, the R_1C_1 integration network provides a 90° phase lag, and the crystal's impedance which is a pure resistance at series resonance together with C_2 acts as a second integration network which provides another 90° phase lag. The time constants of the two RC phase shifting networks should be made as big as possible. This makes their phase shifts independent of any changes in resistance or capacitance values. However, big RC values introduce large gain losses and the selected amplifier should provide sufficient gain to satisfy gain requirement. CMOS inverters or discrete transistors can be used as amplifiers. An experimental evaluation of crystal oscillators using either type of amplifier is given within this report.

CRYSTAL OSCILLATORS USING CMOS-IC

The use of CMOS-IC's in crystal oscillators is quite popular. However, they are not perfect and could cause problems. The input characteristics of such IC's are good, but they are limited in their output drive capability.

The other disadvantage is the longer time delay in a CMOS-inverter as compared to a discrete transistor. The longer this time delay the more power will be dissipated. This time delay is also different among different manufacturers.

As a characteristic of most CMOS-IC's the frequency sensitivity to power supply voltage changes is high. As a group, IC's do not perform very well when compared with discrete transistor circuits.

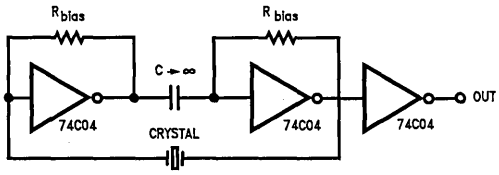
But let us not be discouraged. Low component count which leads to low cost is one good feature of IC oscillators.

As a rule, IC's work best at the low end of their frequency range and poorest at the high end.

Several types of crystal oscillators using CMOS-IC's have been found to work satisfactorily in some applications.

CMOS—TWO INVERTER OSCILLATOR

The two inverter circuit shown in *Figure 2* is a popular one. The circuit is series resonant and uses two cascaded inverters for an amplifier.



TL/DD/8439-2

FIGURE 2

Each inverter has a DC biasing resistor which biases the inverter halfway between the logic "1" and "0" states. This will help the inverters to amplify when the power is applied and the crystal will start oscillation.

The 74C family works better as compared with other CMOS-IC's. Will oscillate at a higher frequency and is less sensitive to temperature changes. The CMOS-COPS data sheet states that a crystal oscillator will typically draw 100 μA more than an external clock source. However, the crystal oscillator described above will draw approximately as much current as an external squarewave clock. The experimental data presented below shows the comparison:

Chip held in Reset, $V_{CC} = +5.0V$

f = 455 kHz, COP444C, CKI is ÷8

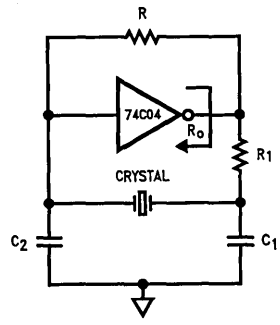
Instruction cycle time = 17.5 μs

I = total power supply (V_{CC}) current drain

Oscillator Type	I (current drain)
Crystal Osc. (data sheet)	950 μA
Crystal Osc. (two inverter)	810 μA
Ext. Clock	790 μA

PIERCE IC OSCILLATOR

Figure 3 shows a Pierce oscillator using CMOS inverter as an amplifier.



TL/DD/8439-3

FIGURE 3

The gain of CMOS inverter is low, so the resistor R_1 should be made small. This reduces gain losses. The output resistance of the inverter (Ro) can be the integrating resistor for the RoC_l phase lag network.

Omitting R_1 or with a small value of R_1, the crystal will be driven at a much higher voltage level. This will increase power dissipation.

For lower frequencies (i.e., 32 kHz), R_1 must be large enough so that the inverter won't overdrive the crystal. Also, if R_1 is too large we won't get an adequate signal back at the inverter's input to maintain oscillation. With large values of R_1 the inverter will remain in its linear region longer and will cause more power dissipation. Typically for 32 kHz, R_1 should be constrained by the relation.

$$\frac{1}{2\pi R_1 C_1} \ll 32\ \text{kHz}$$

At higher frequencies, selection of R_1 is again critical. In order to drive a heavy load at high frequency, the amplifier output impedance must be low. In order to isolate the oscillator output from C_1 so it can drive the following logic stages, then R_1 should be large. But again, R_1 must not be too large, otherwise it will reduce the loop gain.

The value of R_1 is chosen to be roughly equal to the capacitive reactance of C_1 at the frequency of operation, or the value of load impedance Z_L.

$$\text{Where } Z_L = \frac{X_{c1}^2}{R_L}$$

$R_L = R_S$ = series resistance of crystal

The small values of C_1 and C_2 will help minimize the gain reduction they introduce.

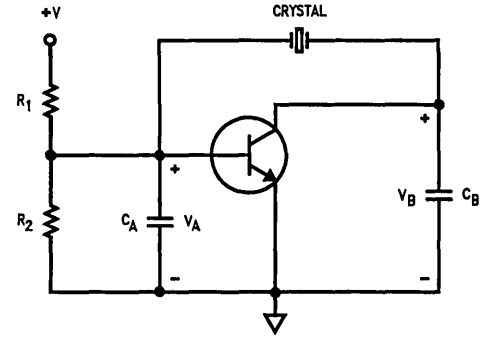
typically: $C_1 = C_2 = 220$ pF at 1 MHz

$C_1 = C_2 = 330$ pF at 2 MHz

DISCRETE TRANSISTOR OSCILLATOR

As mentioned earlier, a discrete transistor circuit performs better than an IC circuit. The reason for this is that in a discrete transistor circuit it is easier to control the crystal's source and load resistances, the gain and signal amplitude.

A discrete transistor circuit has shorter time delay, because it uses one or two transistors. This time delay should always be minimized, since it causes more power dissipation and shifts frequency with temperature changes. *Figure 4* shows a basic Pierce oscillator using a transistor as an amplifier.



TL/DD/8439-4

FIGURE 4

The basic phase shift network consists of C_{A_1}, C_{B_2} and the crystal which looks inductive and is series resonant with C_{A_1} and C_{B_1}. The phase shift through the transistor is 180° and the total phase shift around the loop is 360°. The condition of a unity loop gain must also be satisfied.

$$\frac{V_A}{V_B} = -\left(\frac{C_B}{C_A}\right)$$

$$\frac{V_A}{V_B} = -\left(\frac{X_{CA}}{X_{CB}}\right)$$

For oscillation to occur, the transistor gain must satisfy the relation

$$G\left(\frac{V_A}{V_B}\right) \geq 1$$

where $G = -g_{fe}Z_L$

g_{fe} is the transconductance of the transistor

Z_L is the load seen by the collector

$$Z_L = \frac{X_B^2}{Re}, \quad X_B = -\frac{1}{WC_B}$$

Re is the crystal's effective series resistance.

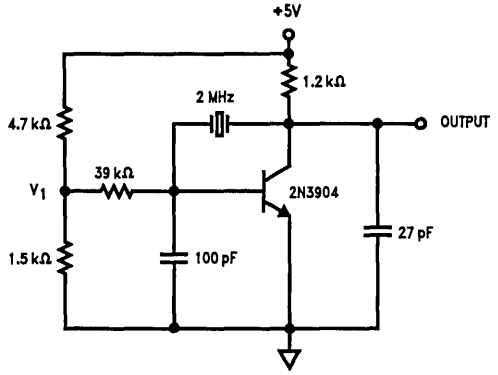
The crystal's drive level

$$P_d = \frac{V_{B2}Re}{X_{B2}}$$

This drive level should not exceed the manufacturer's spec.

Certain biasing conditions might cause collector saturation. Collector saturation increases oscillator's dependence on the supply voltage and should be avoided.

The circuit of *Figure 5* has been tested and has a very good performance.



TL/DD/8439-5

FIGURE 5

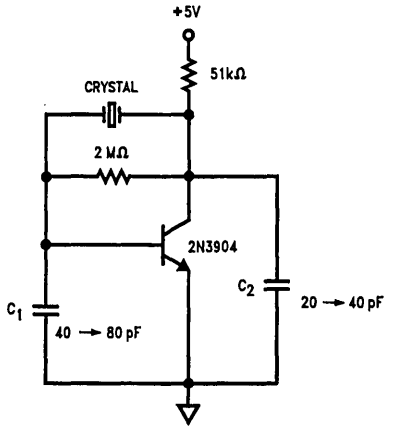
This circuit will oscillate over a wide range of frequencies 2–20 MHz.

$$\text{Voltage } (V_1) = \frac{(5)\,(1.5)}{1.5 + 4.7} = 1.21\text{V}$$

$$\text{Base Current} = \frac{1.21\text{-}V_{BE}}{39\text{k}} = 15.6\ \mu\text{A}$$

At Saturation ($V_{CE} = 0$)

$$I_{C\,(SAT)} = \frac{5}{1.2} = 4.2\text{ mA}$$



TL/DD/8439-6

FIGURE 6

Having 15.6 μA of base current, for saturation to occur

$$h_{FE} = \frac{4.2\ mA}{15.6\ \mu A} = 269$$

The DC beta for 3904 at 1 mA is 70 to 210, so no problem with saturation, even at lower supply voltages.

The current consumption (power supply V_{CC} current drain) of COP444C using the above oscillation circuit is around 267 μA.

The circuit of *Figure 6* is another configuration of discrete transistor oscillator.

The performance of above circuit is also good. The only drawback is that it does not provide larger output signal.

CONCLUSION

As discussed within this report, a discrete transistor circuit gives better performance than an IC circuit. However, oscillators using discrete transistors are more expensive than those using IC's when assembly labor costs are included. So, the selection of either circuit is a trade-off between better performance and cost.

The data and circuits presented here are intended to be used only as a guide for the designer. The networks described are generally simple and inexpensive and have all been observed to be functional. They only provide greater flexibility in the oscillator selection for CMOS-COPS systems.

Selecting Input/Output Options On COPS™ Microcontrollers

National Semiconductor
Application Note 401
Abdul Aleaf

INTRODUCTION

There are a variety of user selectable input and output options available on COPS when the ROM is masked. These options are available to help the user tailor the I/O characteristics of the Microcontroller to the application. This application note is intended to provide the user a guide to the options: What are they? When and how to use which ones? The paper is generally written without reference to a specific device except when examples are given. It must be remembered that any given generic COPS Microcontroller has a subset of all the possible options available and that a given pin might not have all possible options. A reference to the device data sheet will determine which options are available for a specific device and a specific pin of that device.

INPUT/OUTPUT OPTIONS

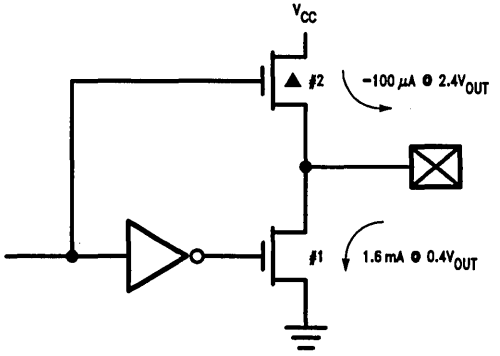
Table I summarizes the I/O capability of NMOS-COPS, in general. However, some of the options have different configuration in CMOS-COPS. Data sheets provide information on the I/O options associated with the CMOS-COPS.

I. OUTPUTS

The following discussion provides detailed information on the capabilities of the mask-programmable output options available on COPS.

A. STANDARD OUTPUT

This option is a simple, straightforward, logic compatible output used for simple logic interface. It is available on SO, SK and all D and G outputs, It is recommended to be used as a default option for all but SO, SK outputs.

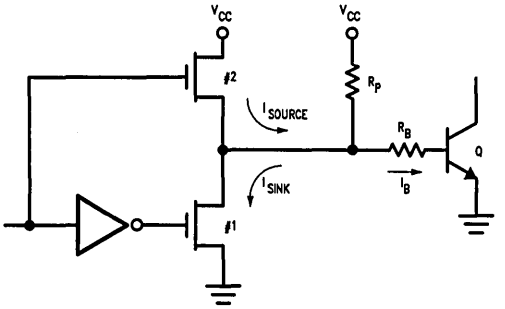


TL/DD/8440-1

FIGURE 1. Standard Output

Figure 1 shows the standard output configuration. The enhancement mode device to ground is good at sinking current (sinks 1–2 mA) and is compatible with the sinking requirement of 1 TTL load (1.6 mA at 0.4V). It will meet the "low" voltage requirement of CMOS logic. All output options use this device (device #1) for current sinking. On the other hand, the relatively high impedance depletion-mode device (device #2) to V_{CC} provides low current sourcing capability (100 μA at 2.4V). This pullup is sufficient to provide the source current for a TTL high level and will go to V_{CC} to meet the "high" voltage requirements of CMOS logic. An external resistor to V_{CC} may be required to interface to other external devices requiring higher sourcing capability.

An interface example to a common emitter NPN transistor is given below:



TL/DD/8440-2

FIGURE 2

R_B is needed to limit transistor's base current if $I_{source} > I_{B(max)}$.

R_p helps generate base drive if the I_{source} is not sufficient. The disadvantage of R_p is the introduction of more power dissipation. The temperature effects on the reverse saturation current I_{CBO} causes I_C to shift. I_{CBO} approximately doubles for every 10°C temperature rise. The effect of changes in I_{CBO} reduces off state margin and increases power dissipation in the off state.

However, in a typical device, the current supplied by R_p will swamp out any effects on I_{CBO}. Another parameter found to be decreasing linearly with temperature is V_{BE}:

$$\Delta V_{BE} = V_{BE_2} - V_{BE_1} = -k(T_2 - T_1)$$

where $k \approx 2$ mV/°C, T in °C.

Now let's consider a practical example:

LOW SOURCE CURRENT OUTPUT:

Standard output, COP420, device #2.

The selected transistor is 2N3904.

DESIGN CONSIDERATIONS:

a. Q is in saturation during ON-state.

b. Q's collector current I_C = 100 mA

TABLE I

	Default	Standard	Push-Pull	High Sink	Very High Sink	LED	Hi-Current LED	TRI-STATE® Push-Pull	Hi Current TRI-STATE Push-Pull	Open Drain
SO	Push-Pull	Logic Compatible; Non MICROWIRE™	MICROWIRE Higher Drive, Faster X'sition							External Pull Up
SK	Push-Pull	Logic Compatible; Non MICROWIRE	MICROWIRE Higher Drive Faster Transition							External Pull Up
D	Standard	Logic Compatible		L Parts Only 15 mA	L Parts Only 30 mA					External Pull Up, Standard, Hi Sink or V.H.S. Pull Down
G	Standard	Logic Compatible; Inputs		L Parts Only 15 mA	L Parts Only 30 mA					External Pull-Up, Standard, Hi Sink or V.H.S. Pull Down
L	Standard	Logic Compatible; Inputs, TRI-LEVEL				Hi Source 1.5 mA TRI-LEVEL	L Parts Only Higher Source 3 mA TRI-LEVEL	MICROBUS™ Meets TRI-STATE Spec. TRI-LEVEL	L Parts Only Meets TRI-STATE Spec. TRI-LEVEL	External Pull Up TRI-LEVEL
H	Standard	Logic Compatible Inputs								External Pull Up
R	Standard	Logic Compatible; Inputs, TRI-LEVEL	Higher Drive Faster Transition TRI-LEVEL					Meets TRI-STATE Spec TRI-LEVEL		External Pull Up TRI-LEVEL

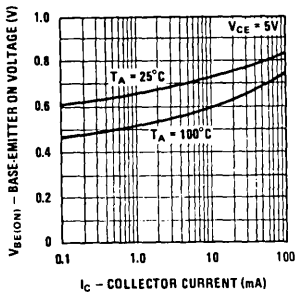
c. Assuming a "forced" of 10 for Q. This is a standard value for β to insure saturation.

For an I_C = 100 mA, β = 10, we have $I_B \geq 10$ mA. The low current standard output certainly cannot provide $I_B \geq 10$ mA. Therefore, a pullup resistor (R_p) is required.

d. Now we need to select the minimum allowed value for R_p. The sinking ability of COPS output will determine R_p. We must sink the pullup current to a $V_{OUT} < V_{BE}$ in order to hold Q off. Also, note that

$$\frac{\Delta V_{BE}}{\Delta T} = -2\ \text{mV/°C}.$$

e. Assuming the worst case is at V_{CC} (max) and High-temperature (let $\Delta T = 20°C \Rightarrow \Delta V_{BE} = -40$ mV). From $V_{BE(ON)}$ Vs. I_C curve, *Figure 3*:



TL/DD/8440-3

FIGURE 3. 2N3904 I/V

at 100 mA, 25°C, $V_{BE} \cong 0.85$V.

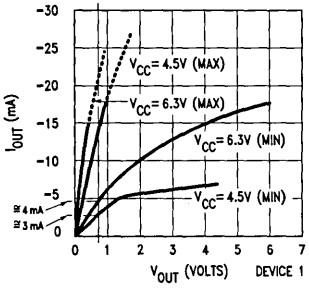
So, our $V_{BE(45°C)} = 0.85 - 0.04 \cong 0.81$V.

There is not margin here for process V_{BE} variations so we can allow 200 mV of slope,

$$V_{BE} = 0.61\text{V (worst case)}$$

f. Having $V_{BE} = 0.61$V, we go to COPS sink graph and draw a vertical line at $V_{OUT} = V_{BE} = 0.61$V. *Figure 4* below:

Output Sink Current



TL/DD/8440-4

FIGURE 4

This will tell us, at $V_{out} = V_{BE}$, how much current can be sinked to keep Q "OFF". The intersection of V_{CC} = 6.3 (MIN) and V_{BE} = 0.61V gives us I_{sink} = 4 mA.

g. Now calculate R_p.

$$R_p \geq \frac{6.3 - 0.61}{4}\text{k} \geq 1.42\text{k}$$

$$\text{the actual standard } R_p\ (\pm 10\%) = \frac{1.42}{0.9}$$

$$= 1.6\text{k} \pm 10\%$$

h. Using the value of R_p, let's calculate the current through R_p at V_{CC} = 4.5V(MIN).

$$I_{R_P} = \frac{4.5 - 0.61}{1.42}\text{mA} = 2.74\text{ mA}$$

Which is less than sink ability of device (3 mA from *Figure 4*) at V_{CC} = 4.5V, V_{out} = 0.61V.

i. Now calculate the available source current. Here we use $V_{BE(max)}$ which is the worst case, and low temperature.

Let T (ambient) = 10°C.

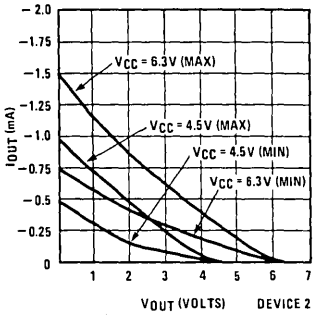
From V_{BE} vs. I_C curve, *Figure 3*:

$V_{BE} \cong 0.83$V at 25°C

$V_{BE} \cong 0.83 + 2\ \text{mv/°C} \times 15 = 0.86$V at 10°C.

Using this value of V_{BE}, we go to COP420 Standard Output source current curve (*Figure 5*), and draw a vertical line at V_{BE} = 0.86V. The intersection of this line and V_{CC} = 4.5(MIN) gives an $I_{source} = 325\ \mu$A.

Standard Output
Source Current



TL/DD/8440-5

FIGURE 5

This is low but typical of N-channel low current standard output.

Contribution of R_p

$$I_{R_P} = \frac{4.5 - 0.86}{\underbrace{(1.6)\,(1.1)}_{R_{p(max)}}} = 2.07\text{ mA}$$

$I_B(\text{min}) \cong 2.07 + 0.325 = 2.3$ mA

This is our worst case base drive, but we needed 10 mA.

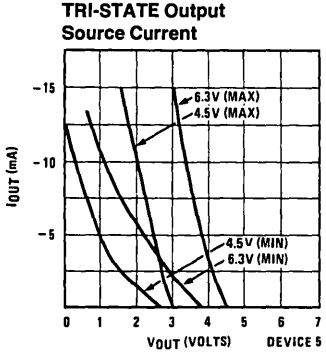
What can we do to get the base drive we need?

1. We can use above design and allow Q to come out of saturation. The disadvantage is that Q's power dissipation increases.

2. Or use a Darlington configuration (Process 05). In such a configuration only first stage of Darlington can be saturated (not output stage). This will introduce a slightly higher power dissipation. Note that for a process 05 transistor, the forced β is 1000.

3. Use a high source type output such as TRI-STATE output. If we draw a vertical line at V_{BE} = 0.86, we get a source current of ≅ 6 mA at V_{CC} = 4.5(MIN) *Figure 6*, which gives us a worst case

$$I_{B(min)} = 8.07 \text{ mA}.$$



TL/DD/8440-6

FIGURE 6

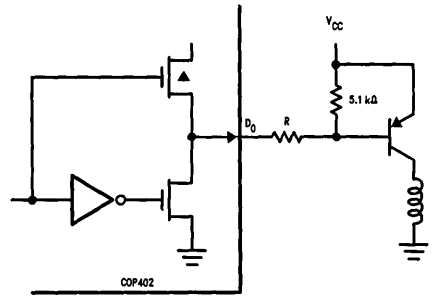
CAUTION On TRI-STATE graph the intersection of $V_{out} = B_{BE}$ = 0.86V and V_{CC} = 6.3V(MAX) curve (*Figure 6*) would result in an $I_{B(Max)}$ = 50–60 mA, which is way too much to handle. In this case there is a need for a series current limiting R_B to kill some of the worst case $I_{B(max)}$.

4. There is a high current Standard-L option on some COPS (i.e., COP4XL, L-port) which provides sufficient source current.

5. N-channel output can generally sink better than source. PNP transistor can be used instead of NPN. The same analysis applies and in general will show better overdirve capabilities.

As shown in *Figure 7*, the D_0 output which has a standard output option, is driving the base of the PNP transistor. Assuming V_{CC} = 4.5V (for COP402), V_{BE} = 1.0V, and a worst case base drive requirement of 3.0 mA. We see that we must supply 200 μA to the base-emitter resistor to turn the transistor on:

$$1.0V/5.1k = 200\ \mu A$$



TL/DD/8440-7

FIGURE 7. PNP Drive

From the output sink current curve on the COP402 data sheet, we find that, at 1.0V the D-line can sink 3.2 mA. To calculate the value of the current limiting resistor,

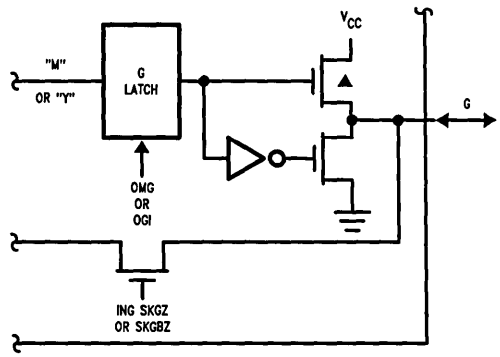
$$R = (V_{CC} - V_{BE} - V_{D0})/I$$

When V_{CC} = 6.3V, the D0 output can sink more than enough current at 0.3V, and if the V_{BE} = 0.7V, we can calculate the maximum D_0 output current:

$$I = (V_{CC} - V_{BE} - V_D)/R$$
$$= (6.3 - 0.7 - 0.3)/780 = 6.3 \text{ mA}.$$

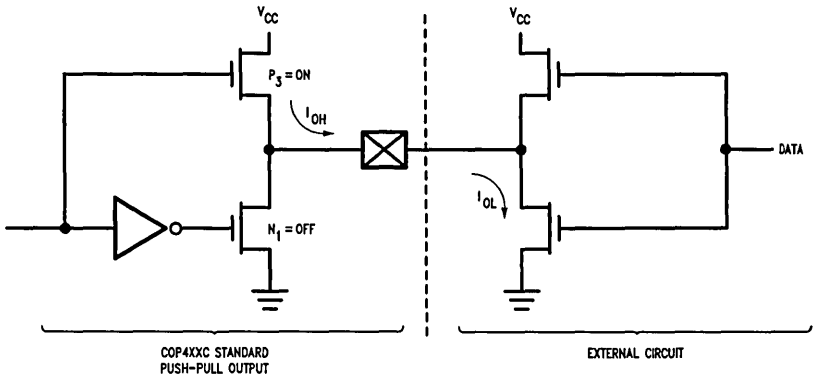
Using the Standard Output Option for Bidirectional I/O (G-port)

The standard output is good at sinking current, but rather weak at sourcing it. Therefore, by using the Standard Drive configuration and outputting 1's to the port, an external source may easily overdrive the port drivers with the added bonus of a built-in pullup. While the depletion-mode device provides sufficient current for a TTL high level, yet can be pulled low by an external source, thus allowing the same pin to be used as an input and output. Data written to the ports is statically latched and remains unchanged until rewritten. As inputs the lines are non-latching (*Figure 8*).



TL/DD/8440-8

FIGURE 8. G Port Characteristics



TL/DD/8440–9

FIGURE 9

When writing a "0" to the port, the enhancement-mode device to ground overcomes the high pullup and provides TTL current sinking capability. While writing a "1" the depletion-mode device behaves as internal pullup maintaining the "1" level indefinitely. In this situation, an input device capable of overriding the small amount of current supplied by the pullup device can be read. This feature provides maximum user flexibility in selecting input/output lines with minimum external components.

In CMOS-COPS the low current push-pull output has even much weaker source current capability and this make it easier to be overriden.

Referring to *Figure 9*.

Note that $I_{OL} > I_{OH}$, otherwise transistors or buffers must be used.

For COP424C/444C, standard push-pull

@ $V_{CC} = 4.5V$, $V_{out} = 0V$, $I_{OH(min)} = 30\ \mu A$
$I_{OH(max)} = 330\ \mu A$
@ $V_{CC} = 2.4V$, $V_{out} = 0V$, $I_{OH(min)} = 6\ \mu A$
$I_{OH(max)} = 80\ \mu A$

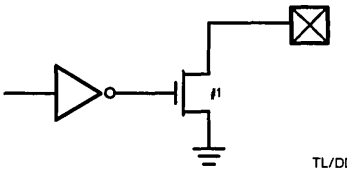
While in NMOS (COP420L), Standard output:

@ $V_{CC} = 4.5V$, $V_{OH} = 2.0V$, $I_{OH(min)} = 30\ \mu A$
$I_{OH(max)} = 250\ \mu A$
@ $V_{CC} = 6.3V$, $V_{OH} = 2.0V$, $I_{OH(min)} = 75\ \mu A$
$I_{OH(max)} = 480\ \mu A$

As we see, both in CMOS and NMOS it is easier to override I_{OH}. Note that the standard output option is available with standard, high, or very high sink current capability ("L" parts only). The pulldown device is bigger for the high/very high current standard output. The sourcing current is the same. These three choices provide some control over current capability.

B. OPEN-DRAIN OUTPUT

This option uses the same enhancement-mode device to ground as the standard output with the same current sinking capability. It does not contain a load device to V_{CC}, allowing external pullup as required by the user's application. The sinking ability of device #1 determines the minimum allowed external pullup. The analysis discussed earlier for Standard Output options equally applies here. Available on SO, SK, and all D, G, and L outputs.



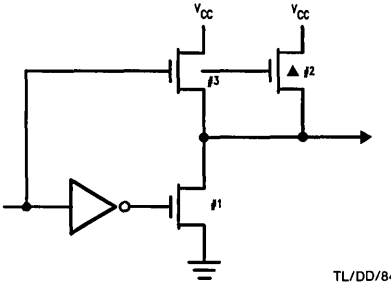
TL/DD/8440–10

FIGURE 10. Open-Drain Output

The open-drain option makes the ports G and L very easy to drive when they are used as inputs. This option is commonly used for high noise margin inputs, unusual logic level inputs as from a diode isolated keyboard, analog channel expansion, and direct capacitive touch-panel interface. Available with standard, high or very high sink capability ("L" parts only).

C. PUSH-PULL OUTPUT

The push-pull output differs from the standard output configuration in having an enhancement-mode device in parallel with the depletion-load device to V_{CC}, providing greater current sourcing capability (better drive) and faster rise and fall times when driving capacitive loads.



TL/DD/8440–11

FIGURE 11. Push-Pull Output

If a push-pull output is interfaced to an external transistor, a current-limiting resistor must be placed in series with the base of the transistor to avoid excessive source current flow out of the push-pull output. This option is generally for MICROWIRE Serial Data exchange.

It is available on SO, SK only and is recommended to be used as a default option for these outputs. A few points must be kept in mind when using SO, SK for MICROWIRE interface.

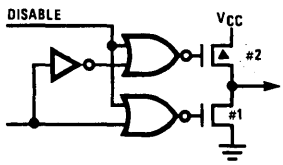
The data sheet specifies the propagation delay for a certain test condition (i.e., V_{CC} = 5V, V_{OH} = 0.4V, Loading = 50 pF, etc.).

In practice, actual delay varies according to actual input capacitive loading (typical 7–10 pF per IC input), total wire capacitance and PCB stray capacitance connected to the SI input. Thus, if actual total capacitive loading is too large to satisfy the delay time relationships ($t_d = t_{SK} - t_r$; t_d = actual delay time, t_{SK} = the instruction cycle time, t_r = the finite SK rise time), either slow down SK cycle time or add a pullup resistor to speed up SK "0" to "1" transition or use an external buffer to drive the large load. Besides the timing requirement, system supply and fan-out/fan-in requirements have to be considered, too.

If devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Briefly, for devices that have incompatible input levels or source/sink requirements to exchange data, external pullups or buffers are necessary to provide level shifting or driving. Unreliable operation might occur during data transfer, otherwise. For a 100 pF load, a standard COPS Microcontroller may use a 4.7k external resistor, with the output "low" level increased by less than 0.2V. For the same load the low power COPS may use a 22k resistor; with the SO, SK output "low" level increased by less than 0.1V.

D. STANDARD L OUTPUT

Same as Standard Output, but may be disabled. Available on L-outputs only.



TL/DD/8440–12

FIGURE 12. Standard L Output

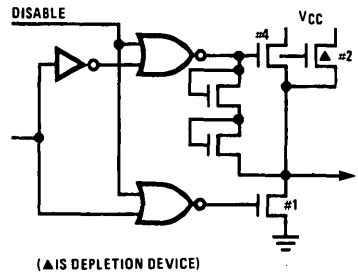
When this option is implemented on the L-port and the L-drivers are disabled to use the L lines as inputs, the disabled depletion-mode device cannot be relied on to source sufficient current to pull an input to a logic high. There are two ways to use L lines as inputs (having standard L option):

The first method requires that the drivers be disabled. In this case the lines are floating in an undefined state. The external circuitry must provide good logic levels both high and low to the input pins. The inputs are then read by the INL instruction. The second method is similar to the technique used for the G-port. The drivers are enabled and a"1" must be written to the Q register.

The external circuitry will then be required only to pull the lines low to a logic "0". The line will pull up to a "1" itself. The INL instruction is used as before to read the lines.

E. LED DIRECT DRIVE OUTPUT

In this configuration, the depletion-load device to V_{CC} is paralleled by an enhancement-mode device to V_{CC} to allow for the greater current sourcing capacity required by the segments of an LED display. Source current is clamped to prevent excessive source current flow.



TL/DD/8440–13

FIGURE 13. LED (L output) NMOS-COPS

This configuration can be disabled under program control by resetting bit 2 (EN^2) of the enable register to provide simplified display segment blanking.

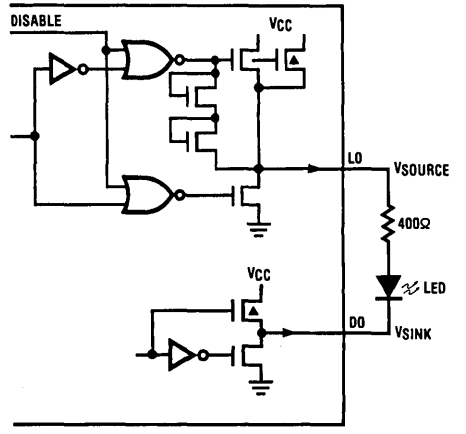
However, while both enhancement-mode devices are turned off in the disabled mode, the depletion-load device to V_{CC} will still source up to 0.125 mA. As in the case of Standard L output, again this current is not sufficient to pull an input to a logic "1".

The drivers must be disabled and the lines must be pulled high and low externally, whenever they are used as inputs.

Example #1:

When COPS outputs are used to drive loads directly, the power consumed in the outputs must be considered in the maximum power dissipation of the package.

Figure 14 shows an LED segment obtaining its source current from L_0 output and D_0 sinking the current. In this configuration all the power required to drive the LED with the exception of the portion consumed by the LED itself, is consumed within the chip. Assuming COP404L is the driving device:



TL/DD/8440–14

FIGURE 14. LED Drive

If we assume the V_{source} is not inserted, the device has a V_{CC} of 9.5V, and that the voltage drop across the LED is 2.0V.

We can calculate the power dissipation in these outputs. The minimum current that D_0 can sink at 1.0V is 35 mA (COP404L data sheet). L_0 can source up to 35 mA at 3.0V. Therefore, the power dissipation for the L_0 output could be: (9.5 −3.0) (0.035) = 227 mW. The power in the D_0 output is (1)(0.035) = 35 mW.

Now let us calculate the current limiting resistor. Referring to COP404L L_0–L_7 output source current curves, at V_{CC} =9.5V the minimum current curve peaks at I = 6.0 mA and V_{source} = 4.8V. The current curve is actually very flat between 4.0 and 5.0 volts. For maximum current, we need to set the voltage on the L pin equal to 4.8V at 6.0 mA. The D line will sink this current at 0.4V. Therefore, the resistor and LED must make up the difference:

$$V_I = V_D + IR + V_{LED}$$
$$4.8 = 0.4 + 0.006R + 2.0$$
$$R = 400\Omega$$

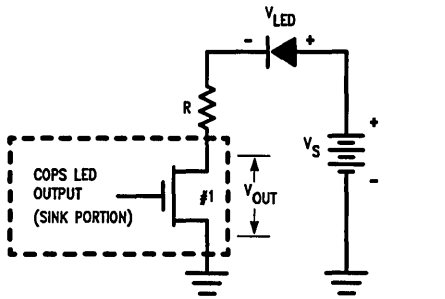
At the other end of the curve, when the L line sources the maximum current, assume the LED and the D line will have the same voltage drop.

$$V_I = 0.4 + IR + 2.0$$
$$V_I = 2.4 + IR$$

From the current curve, we see that at 6.4V the L line will source 10 mA. Therefore: V_I = 2.4 + (0.01) (400) = 6.4V.

Example #2:

Let's consider a different configuration.



TL/DD/8440-15

FIGURE 15. LED DRIVE

Now we calculate the series current limiting resistor R. The circuit has two non-linear devices to be considered; the output device and the LED.

The LED in this example is NSC5050. Looking at I/V curve, the device has a threshold 1.6V. Also, note that for V_{LED} > 1.6V the I/V curve is very linear (*Figure 17*).

Because of this, the LED characteristic can be modeled as a sharp threshold device with a non-zero source resistance (normally I/V curve is LOG looking).

From ON part of curve,

$$R_S = \frac{1.9 - 1.7}{0.05} = 4\Omega$$

We can neglect R_S as well (only $R_S \ll R$). Our model is simply a voltage source for the LED when

$I = 0$ for $V_{LED} < V_{TH}$

$I = \infty$ for $V_{LED} > V_{TH}$

Design Procedure:

1. $$I_{LED(min)} = \frac{V_{S(min)} - (V_{LED(max)} + V_{OUT(max)})}{R(max)}$$

We need endpoints of the load line.

a. @V_{out} = 0 $\Rightarrow$ $$I_{LED(min)} = \frac{V_{S(min)} - V_{LED(max)}}{R(max)}$$

b. @$V_{out} + V_{LED(max)} = V_S \Rightarrow I = 0$
($V_{LED(max)}$ = 2V)

2. Plot a and b

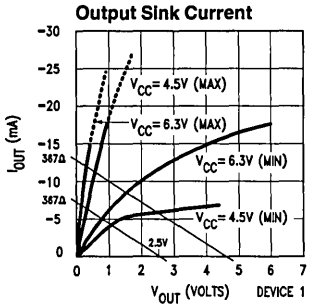
Assuming an I_{min} = 7 mA, $V_{S(min)}$ = 4.5V

from 1 $R_{(max)}$ = 357Ω

Draw the load line with slope −1/357 crossing

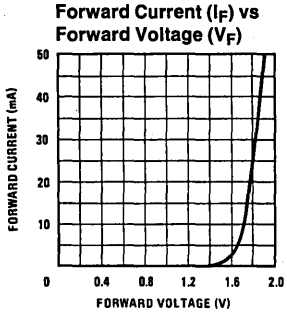
$V_{out} = V_S - V_{LED(max)} = 4.5 - 2 = 2.5V$.

(*Figure 16*).



TL/DD/8440-16

FIGURE 16. COP420



TL/DD/8440-17

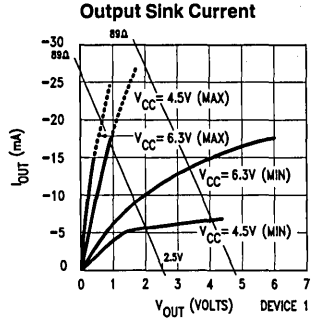
FIGURE 17. LED I/V Characteristic

The intersection of this load line and V_{CC} = 4.5V (min) curve, we find an actual value of $I_{(min)}$ = 4.25 mA.

To determine I_{max} (at R = 357Ω) we draw a parallel load line intersecting V_{out} = 6.3 − 2.0 = 4.3V and find that @ V_{CC} = 6.3V, $I_{(max)}$ = 13 mA.

3. From above calculations we observe that our $I_{(min)}$ (actual) is way off. Let's try to rotate our first load line around V_{out} = 2.5V to increase I_{min} and then check I_{max} and R. (*Figure 18*).

Let's go for an I_{min} (actual) = 6 mA. This will give us R = 89Ω and the max. plot goes off the graph to = 36 mA.



TL/DD/8440-18

FIGURE 18. COP420

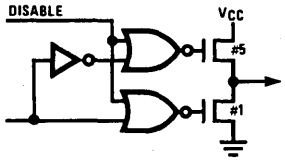
Comments:

1. The design must be a compromise between the two extremes (battery life should also be considered).

2. The lower the LED threshold the better. (The load line moves further up the device curve.)

F. TRI-STATE PUSH-PULL OUTPUT

This option is specifically available to meet the specifications of National's MICROBUS, outputting data over the data bus to a host CPU. It has two enhancement-mode devices to ground and V_{CC}.



TL/DD/8440-19

FIGURE 19. TRI-STATE Push Pull (L output)

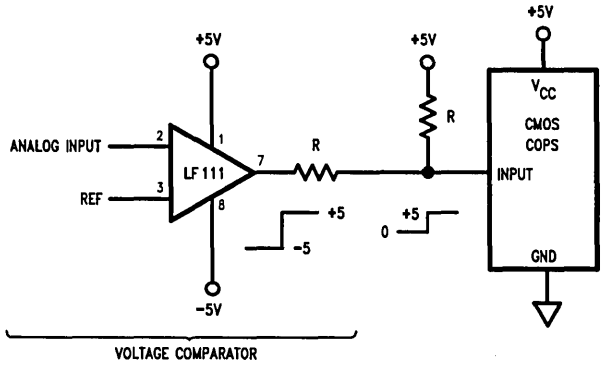
The TRI-STATE logic can disable both enhancement-mode devices to free the MICROBUS data lines for input operation.

CAUTION Never try to pull against the TRI-STATE Output (too much source current) with the drivers enabled and Q register previously loaded with "1". The choices we have are mentioned earlier. Either TRI-STATE L-port or use Standard L output option.

II. INPUTS

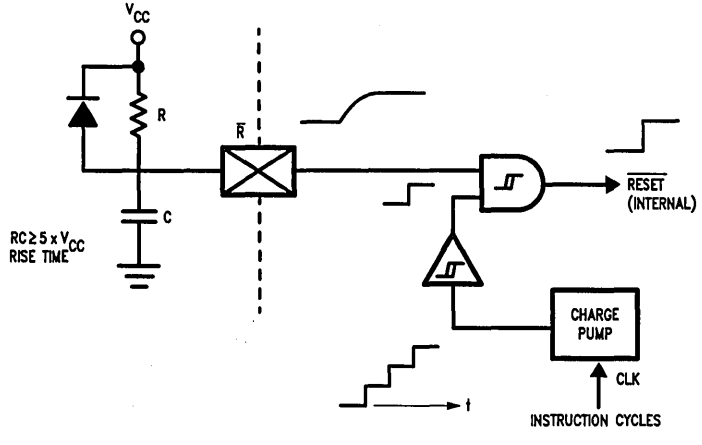
COPS inputs may be programmed either with a depletion load device to V_{CC} or floating (Hi-Z input). All inputs are TTL/CMOS compatible. Hi-Z inputs should not be left floating; they should be connected to the output of a "high" and "low" driving device if active or to V_{CC} and ground if unused. Especially when using CMOS COPS (very high impedance inputs), the open inputs can float to any voltage. This will cause incorrect logic function and more power dissipation. Also, the CMOS inputs are more susceptible to static charge which causes gate oxide rupture and destroys the device. Unlike inputs, the outputs should be left open to allow the output switch without drawing any DC current. Another precaution is powering up the device. Never apply power to inputs or TRI-STATE outputs before both V_{CC} and ground are connected. This will forward bias input protection diodes, causing excessive diode currents. It will also power the device.

Special care must be practiced when interfacing a CMOS-COPS input to an analog IC, powered by different supply voltages. Avoid overvoltage conditions resulting



TL/DD/8440-20

FIGURE 20



TL/DD/8440-21

FIGURE 21

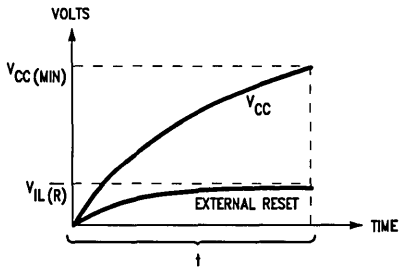
from such situations. As an example, consider the interface of a CMOS-COPS with the LF111 voltage comparator:

When the low level "−5V" appears on the comparator's output, the COPS input is pulled low below "logic low" of "0V". This will cause damage if the comparator sinks enough current. The use of a current-limiting resistor in series with the input is helpful. A better solution is to use a voltage divider as shown in *Figure 20*. Any time a low level appears on the comparator's output, a total voltage drop of 10V will appear across both resistors each dropping 5V, causing the input to sit at 0V. Whenever the output goes high, the resistors will not drop any voltage (no current through the resistors) and a logic high of 5V will appear on the input. To reduce power dissipation introduced by resistors, the resistor value must be high (>100k), because the CMOS inputs have very high input impedance.

$\overline{\text{RESET}}$ INPUT

All COPS Microcontroller have internal reset circuitry. Internally there is an AND gate with one input coming from the $\overline{\text{RESET}}$ input, and the second input connected to a charge pump circuitry. In the Charge pump circuit, a tiny capacitor is being charged upon execution of each internal instruction cycle. When the voltage across this internal capacitor reaches a high logic level, the second input of the AND gate is released.

The Reset logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. With a slowly rising power supply, the part may start running before V_{CC} is within the guaranteed range. In this case, the user must provide an external RC network and diode shown in *Figure 21* above. The external RC network is there to hold the $\overline{\text{RESET}}$ pin below V_{IL} until V_{CC} reaches at least $V_{CC(min)}$. The desired response is shown in *Figure 22*.



TL/DD/8440-22

FIGURE 22

t = 500-600 instruction cycles (8 msec)
for COPxxxL

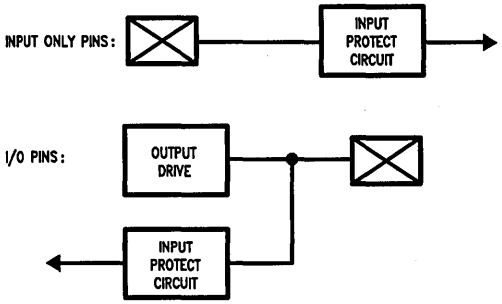
t = 900–1000 instruction cycles (4 msec)
for COPxxxC

The diode is included in the reset circuitry to cause a "forced Reset" when the power supply goes away and recovers quickly. In such a situation the diode helps discharge the capacitor quickly. Otherwise, if the power failure occurs for a short time, the capacitor will not be fully discharged and the chip will continue operation with incorrect data.

Note that on the CMOS COPS, the internal charge pump circuitry can be disabled when using a very slow clock (<32 kHz) [option 23 = 1]. This is necessary, because one can run from DC to 4 μs instruction cycle time (fully static). In such a situation external RC network discussed earlier must be used.

INPUT PROTECTION DEVICES

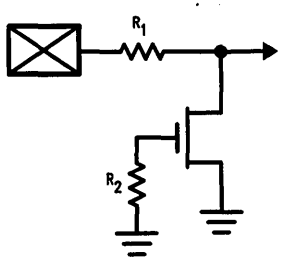
All inputs and I/O pins have input protection circuitry. This circuitry is there regardless of any option selected. It is the first circuitry encountered at the pin.



TL/DD/8440–23

FIGURE 23

For NMOS and XMOS devices, the circuits are of the form:

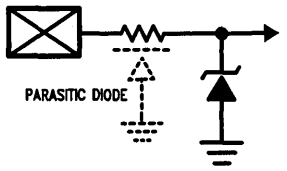


TL/DD/8440–24

FIGURE 24

This is a standard circuit defined for the process. R_1 is on the order of 200Ω. R_2 is around 300Ω (note that the R values are not precise).

This circuit is functionally equivalent to:



TL/DD/8440–25

FIGURE 25

The zener breakdown is around 10–15V; the gate breakdown is 50V.

CONCLUSION

All COPS Microcontrollers have a number of I/O options necessary to implement dedicated control functions in a wide variety of applications. The flexibility to select different options allows the user to tailor within limits, the I/O characteristics of the Microcontroller to the system. Thus, the user can optimize COPS for the system, thereby achieving maximum capability and minimum cost. This application note deals with the basic functionality of COPS I/O characteristics and does not address electrical differences among the various COPS devices.

New CMOS Vacuum Fluorescent Drivers Enable Three Chip System to Provide Intelligent Control of Dot Matrix VF Display

National Semiconductor
Application Note 440
Tom Markman

INTRODUCTION

Vacuum Fluorescent (VF) displays are becoming more and more common in a variety of applications. Manufacturers of everything from Automobiles to Video Recorders have taken advantage of these easy to read displays. VF displays are available in a wide variety of configurations; clock displays, calculator displays, multi-segment, and dot matrix displays are readily available at a low cost. This application note develops and covers in some detail a small CMOS system consisting of a single chip microcontroller and two display drivers which control a 20 character, 5 x 7 dot matrix VF display.

Figure 1 shows the schematic of the system. The microcontroller, a COPS™ 424C, receives a character in ASCII form from the host system, stores the ASCII value of the character in its onboard RAM, converts the ASCII value to a 5 byte data word suitable for the display drivers and displays it on the VF display. The COPS also refreshes the display continuously while performing character update, much like a dumb terminal. Not including the address decoding logic, this application requires only the onboard RAM and ROM of the COPS424C, and National's MM58341 and MM58348 VF display drivers. If a steady message or a scrolling sentence is desired, only small changes in the COPS software are required. In this case the messages could be stored in the ROM of the COPS and the need for a host system would be eliminated.

VF DISPLAY AND VF DISPLAY DRIVER REQUIREMENTS

The display used in this application was an Itron #DC205G2. This 20 segment, 5 x 7 dot matrix, multiplexed display required a filament voltage of 5.7 Vac and a filament current of 37 mAac. The anode and grid voltages were supplied by the display drivers. The voltage and current requirements vary considerably for different displays depending on the size and number of characters, and the configuration (dot matrix, 7 segment, 14 segment, etc.). To determine the voltage requirements for a particular display, a simple calculation can be made. If maximum possible brightness of the display is desired, the following equation must be true:

$E_t \geq E_b + E_k + (I_b)(R_{on})$ where:

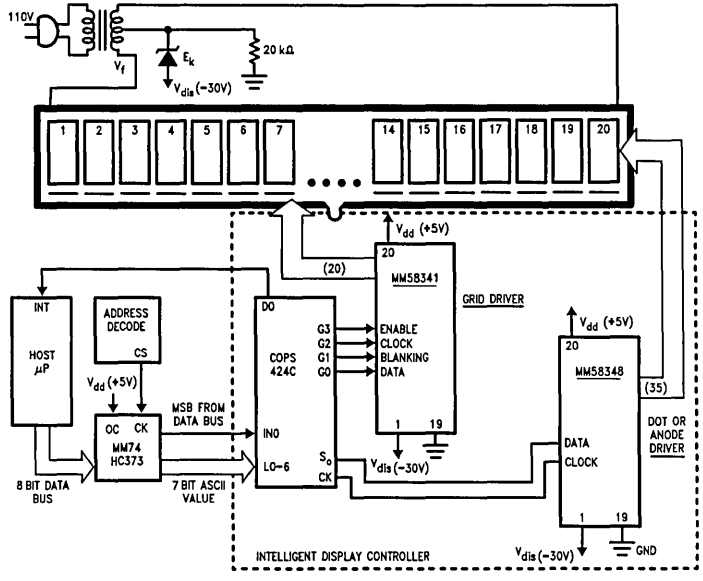
E_t is the total Voltage of the display drvier or $|V_{dis}| + V_{dd}$

E_k is the display Cathode Bias Voltage

$E_b = E_c$ is the typical Anode or Grid Voltage (V_{p-p})

I_b is the typical anode current (mAp-p)

R_{on} is the display driver output impedance (Ω)



TL/F/8683–1

FIGURE 1. System Diagram Showing the Basic 3-Chip Display Controller and the Interface to a Microprocessor System

If the maximum brightness is not desired, the following equation can be used: $(E_t)(1.2) \geq E_b + E_k + (I_b)(R_{on})$. In this application, the calculated E_t was 42.25V, however, the display was legible under normal lighting conditions, with an E_t as low as 25V. If your display requires more than the 35V output of the MM58341 and MM58348, pin for pin compatible 60V VF Display Drivers (MM58241, MM58248) are available.

Figure 2 shows the relationship between the required VF display voltages. The cut-off voltage (E_k) is set by the Zener diode on the center tap of the filament transformer. This value is given in the VF display data sheet.

Avoiding Flicker and Pulsing

There are two different conditions which may cause the display to appear to flicker. The first is the refresh rate. This is particularly a problem on displays where the micro-controller must up-date more than 25 characters. Since the human eye begins to notice flicker at about 40 Hz, a display with a refresh rate less than that will appear to be flashing on and off.

The second type of flicker occurs when the refresh rate is between 40 Hz and 90 Hz. In this case, the display will appear to be rolling rather than flashing. This condition occurs when the refresh rate and the filament frequency are close together. If a character is only on during the time when the filament voltage is negative, it will appear to be slightly brighter than the character next to it which may only be on during the positive cycle of the filament voltage. If this is the case, as it was in this application, the simplest solution is to increase the frequency of the filament. A DC oscillator circuit, such as the one shown in *Figure 3*, can be used to replace the AC voltage source. The filament frequency can be easily adjusted to eliminate this condition.

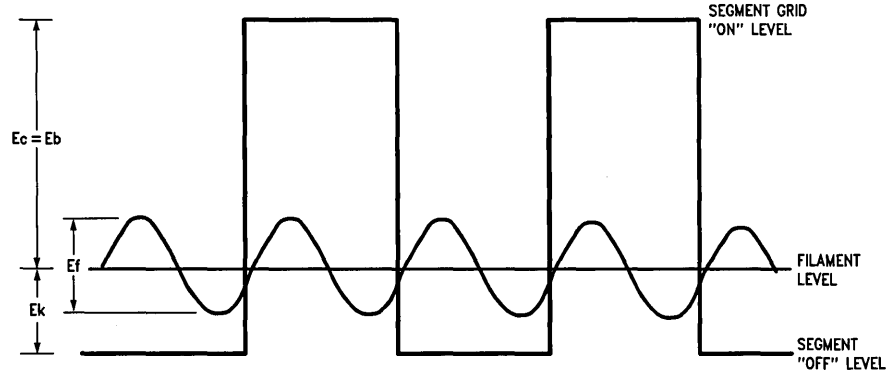


FIGURE 2. Voltage Levels for VF Display

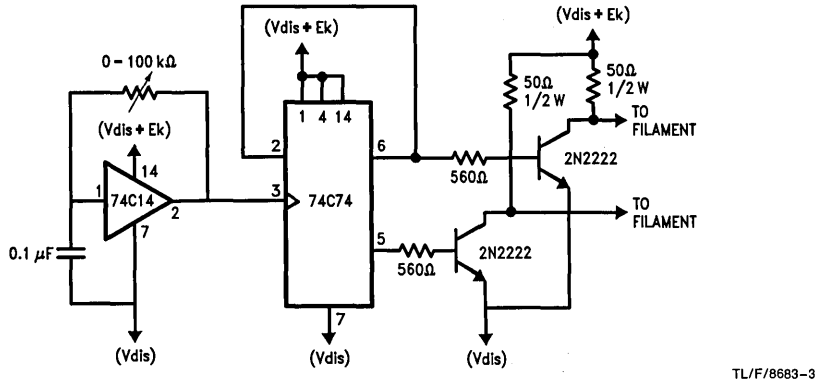


FIGURE 3. Filament Oscillator Circuit

VF Display Drivers

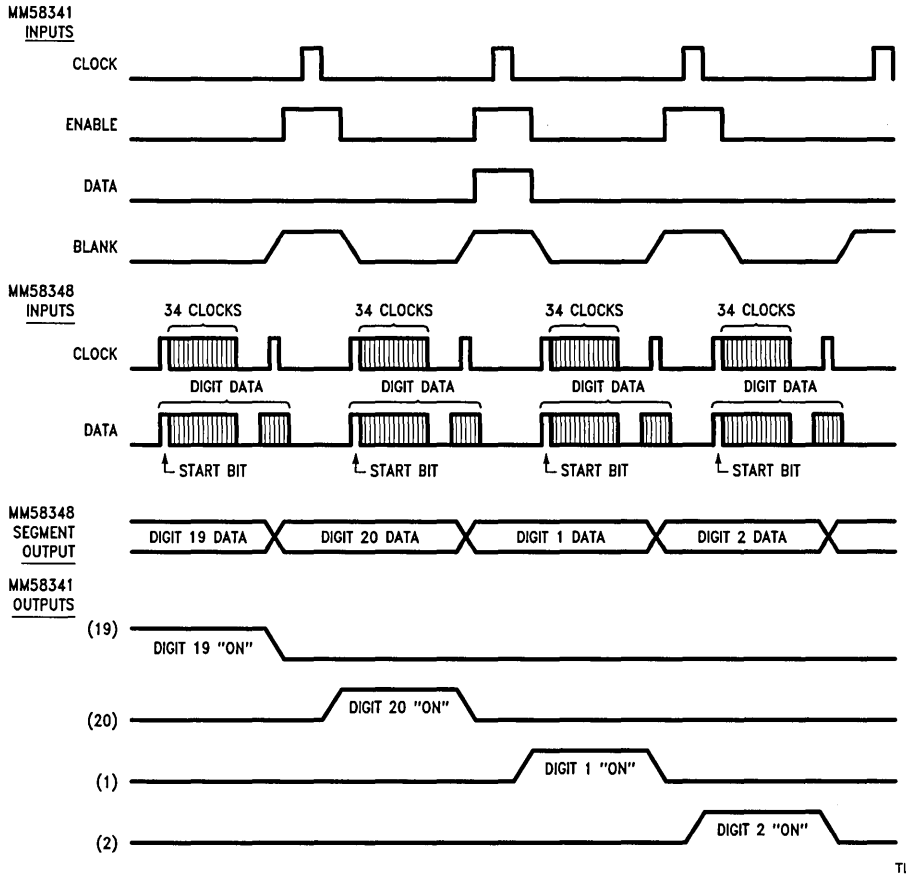
Two high voltage display drivers were needed to control the VF display. A MM58341, was used to control the grids and a MM58348 was used to control the individual pixels or anodes. Both of these drivers receive serial information and output 32 and 35 segments of data respectively.

The MM58341 has three control pins which make it ideal for controlling the grids of a VF display. The blanking control pin will turn off all segments of the display when a logic '1' is applied to this pin. This is particularly important for reducing ghosting, and controlling brightness. Ghosting is a condition where the last characters shadow appears behind the character being displayed. The enable pin acts as an envelope for the input signal. Only while it is at a logic '1' level will the circuit accept clock inputs. When the pin goes low, all the data is latched and displayed. A data out pin is also provided for cascading. If the display has more than 32 grids, a second grid driver can be cascaded by connecting the data out pin to the input data for the second grid driver.

The MM58348 is a 35 bit shift register and latch which is used to control each pixel or dot. When a leading 1, followed by 35 bits of data, is received, the data is latched and displayed. The chip is automatically reset upon power up.

MULTIPLEXED DISPLAY REFRESH TIMING

Considering first the digit driver (MM58341), it becomes clear that the digits must be enabled or refreshed sequentially and that this process must be continuous regardless if the display data has changed. The data for the MM58341 is simply a 1 followed by 19 zeroes where the 1 is shifted through the internal registers of the MM58341. As each digit is enabled, the corresponding segment data is displayed. To insure that no ghosting effects are seen during the transition between digits, the blanking control is activiated just before the data is latched into the dot or anode driver and deactivated just after the data has been latched. During this time when the blanking control is activated, the grid driver is clocked shifting the 1 to the next location. *Figure 4* shows the micro-controller waveforms and the resultant display waveforms for the 20 character display.



TL/F/8683–4

FIGURE 4. Timing Diagram

In between digit strobes, the segment data is updated. The first 34 bits of segment data are set up in the dot driver and the blanking signal is activated to disable all 20 digits. The 35th bit of data is clocked in, updating the segments. Since the MM58348 resets its internal shift register each time the data is latched, it can accept all but the final data bit while still displaying the previous digit. The digit driver is then clocked, shifting the digit strobe to the next position. The enable is then brought low, enabling the next digit. Finally blanking control is deactivated and the data displayed.

During the time which the blanking control is high, the order in which the segments or the digits are updated is not critical. Since this occurs while the display is blank. The digit driver may be clocked first, or the segments could be changed first. In general, the philosophy for the driving this VF multiplexed display is outlined in *Figure 5*.

HOST INTERFACE AND PROGRAMMING

With a minimal amount of address decoding and an eight bit latch, COPS can be interfaced with a common microprocessor bus. When a character has been input into the host to be displayed, the ASCII value of that character is latched into the eight bit latch (MM74HC373) and is read on the L port (L0–6) of the COPS. The MSB of the ASCII value must be a logic 1. This MSB is the signal to the COPS that a new character is being presented. Once the character has been stored, an interrupt is sent from the COPS to the host through the D-0 port. The COPS checks for a new character being input every 200 μs. If a character is being sent, 1 ms is required to store that character in the RAM of the COPS. With the COPS controlling the display, the host micro-processor is not being tied down with character look-up and display refresh. A simple flowchart of the host requirements is shown in *Figure 6*.

COPS SOFTWARE

There are four main sections of the COPS software. The first section, the initialization of the RAM, sets up the RAM as shown in *Figure 7*. A '0' is stored in all of the LSB positions and a '2' is stored in all of the MSB positions. Since the COPS is in a constant display loop, this is necessary to insure a blank display. 20H is the ASCII value of a space. With the RAM set up in this way, a maximum of 28 characters can be stored in RAM. Since the display in this application is only 20 characters long, RAM locations M1,4 to M1,11 and M3,4 to M3,11 are not used. RAM locations 1,12 to 1,15 and 3,12 to 3,15 are used as temporary storage throughout the program and cannot be used for character storage.

The second part of the program, stores the new characters sent by the host CPU in RAM. Once a character has been sent, this section of the program checks the ASCII value of that character to see if it is a control character or a display character. If it is a display character, the character is stored in RAM and an interrupt is sent to the host. There are three control characters which the COPS program will recognize. Cursor forward (ASCII value 08H) moves the cursor forward without destroying the data, cursor backwards (ASCII value 0CH) moves the cursor backwards without destroying the data, and return (ASCII value 0DH) will clear the display and put the cursor at the beginning of the display. To recognize and store a character, 1 ms is required.

The third part of the program, the display loop, is the heart of the program. Unless a new character has been detected, the program is always in this loop. This section does the

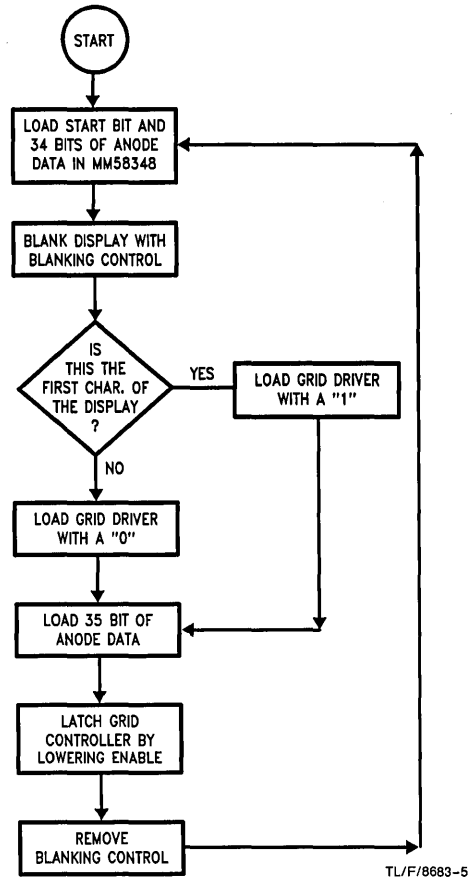


FIGURE 5. Flowchart for Display Drivers

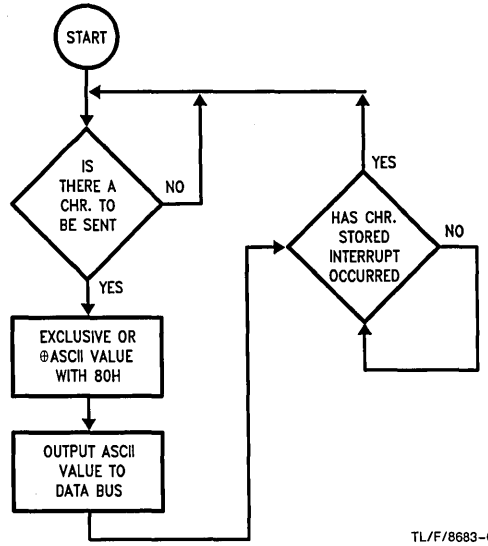


FIGURE 6. Host System Flowchart

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
LSB Chr 1	LSB Chr 2	LSB Chr 3	LSB Chr 4	LSB Chr 5	LSB Chr 6	LSB Chr 7	LSB Chr 8	LSB Chr 9	LSB Chr 10	LSB Chr 11	LSB Chr 12	LSB Chr 13	LSB Chr 14	LSB Chr 15	LSB Chr 16	M0
MSB Pointer	LSB Pointer	Temp. ASCII STORAGE										LSB Chr 17	LSB Chr 18	LSB Chr 19	LSB Chr 20	M1
MSB Chr 1	MSB Chr 2	MSB Chr 3	MSB Chr 4	MSB Chr 5	MSB Chr 6	MSB Chr 7	MSB Chr 8	MSB Chr 9	MSB Chr 10	MSB Chr 11	MSB Chr 12	MSB Chr 13	MSB Chr 14	MSB Chr 15	MSB Chr 16	M2
Temp. Storage of Pointer												MSB Chr 17	MSB Chr 18	MSB Chr 19	MSB Chr 20	M3

FIGURE 7. COPS RAM Map

Matrix	PAD	Column 1	Column 2	Column 3	Column 4	Column 5	PAD
Binary	0 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0						
Hex.	13		EA	24	28	3E	

FIGURE 8

character font look-up, shifts the character data out the COPS serial port to the MM58348, and controls the MM58341 through the four bit parallel port (G0–4). Because the most significant nibble of the program counter is used as part of some COPS instructions, it is important that parts of the program are located at specific locations in ROM.

The final part of the program is the data. Each character is represented by a 5 byte data word. Each byte of the data word is stored at a different location in ROM. Fonts for characters with the ASCII values from 20H–5AH have already been stored in ROM. These characters can be changed or more characters can be added. The only limitation to the number of characters is the amount of available ROM.

CREATING THE 5 BYTE DATA WORD

Any number or combination of pixels or dots can be turned on at a time. To create a new character, it is easiest to first create a binary string which represents the character. A '1' in the binary string will turn on the pixel, a '0' will turn it off. To create this string, start in the upper left corner of the matrix and go down the columns.

The letter 'A' (*Figure 9*) would have a binary string shown in *Figure 8*. The data must be padded to make it an even 5 bytes in length. The pad at the beginning of the data (0001) is used as the leading 1 for the MM58348. The one bit pad at the end of the binary string must be a 0. If a 1 were sent as the pad, it would be used as the start bit for the next character.

The 5 byte data word that would be stored in ROM and represent the letter 'A' would then be 13EA24283E.

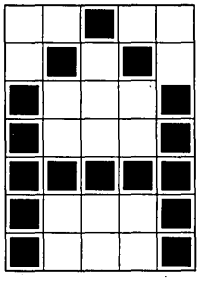
STORING THE DATA IN ROM

The 5 bytes of data are stored in 5 different locations in ROM. The first byte of data will be stored, LSB first, at location 200H plus the ASCII value of the character. For example, the ASCII value of the letter 'A' is 41H. The first byte of data for the letter 'A' would be stored, least significant bit first, at 241H. The second byte of data is stored at the location of the first data byte plus 60H or in this case at 2A1H. The location of the third byte is 40H plus the location of the second byte. In this case, the third byte of data would be stored at 2E1H. The fourth byte of data is stored at 300H plus the ASCII value of the character or at 341H for the letter 'A'. The final byte of data is stored 40H from the fourth byte or at 381H. Remember the LSB of each byte is stored first. Table I shows the locations in ROM and the values stored in them for the letter 'A'.

This application shows a VF display controller designed with a minimum number of IC's. If additional information about VF displays or VF display drivers is required, refer to Application Note AN-371 (The MM58348/342/341/248/242/241 direct drive Vacuum Fluorescent (VF) Displays.

TABLE I. Character Data of 'A' and Its Locations in ROM

Address In ROM	Data Stored
0241H	31
02A1H	AE
02E1H	42
0341H	82
0381H	E3



TL/F/8683–7

FIGURE 9. 5 x 7 Character as Stored in ROM

Section 1 of COPS Software

```
        .CHIP 424C          ;DEFINES COPS CHIP
;THIS SECTION INITIALIZES THE RAM IN THE COPS BY LOADING A
;2 IN THE MSB AND A 0 IN THE LSB LOCATIONS OF EACH CHARACTER.
;IT ALSO STOPS THE CLOCK AND SETS THE POINTER AT THE FIRST
;CHARACTER OF THE DISPLAY.

RESET:          CLRA
                LBI 3,15        ;LOADS A 2 IN ALL
                JSR CLEAR2      ;MSB LOCATIONS
                LBI 2,15        ;LOADS A 2 IN ALL
                JSR CLEAR2      ;MSB LOCATIONS
                LBI 1,15        ;LOADS A 0 IN ALL
                JSR CLEAR       ;LSB LOCATIONS
                LBI 0,15        ;LOADS A 0 IN ALL
                JSR CLEAR       ;LSB LOCATIONS

                CLRA            ;LOADS POINTER IN RAM
                XAD 1,15        ;MSB IN 1,0F
                CLRA
                AISC 15         ;LSB IN 1,0E
                XAD 1,14

                RC              ;RESETS CARRY TO
                XAS             ;    STOP CLOCK
                JMP START

CLEAR:          CLRA            ;CLEARS REGISTORS
                XDS 0
                JMP CLEAR
                RET

CLEAR2:         CLRA            ;PUTS A 2 IN REGISTORS
                AISC 02
                XDS 0
                JMP CLEAR2
                RET
```

Section 2 of COPS Software

```
;THIS SECTION OF CODE IS ONLY EXECUTED WHEN A NEW
;CHARACTER HAS BEEN ENTERED. IF THE CHARACTER IS
;A CONTROL CHARACTER, THE CURSOR IS MOVED ACCORDINGLY,
;OTHERWISE THE CHARACTER IS STORED IN THE RAM OF THE COPS.

          ;NEW CHARACTER HAS BEEN ENTERED
NEW:            LBI 1,0C        ;DUMMY POINTER
                INL             ;READS ASCII FROM
                XIS  0          ;DATA BUS
                X 0
                LDD 1,0D
                RC              ;CHAR.  MSB=0 THEN YES
                AISC 15         ;MSB<>0 THEN NO
                JMP SPECIAL
                AISC 01
                LDD 1,0E        ;STORE ASCII IN RAM
                CAB
                LDD 1,0F
                XABR
                LDD 1,0C        ;MSB IN 1,0C
                X 2
                LDD 1,0D        ;LSB IN 1, 0D
                X 0
```

Section 2 of COPS Software (Continued)

```
                JSR CURFOR
                LBI 0,01        ;SENDS INTERRUPT TO
                OBD             ;  HOST.  CHAR. IS
                LBI 0,0         ;  STORED IN RAM
                OBD
                JMP START

            ;SPECIAL CHARS. (CR, LF, CLEAR DISPLAY)

CURFOR:         LDD 1,0E        ;MOVES CURSOR FORWARD ONE
                COMP            ;SPACE.  IF CURSOR IS
                AISC 01         ;MOVED BEYOND THE END OF
                JMP OK          ;DISPLAY, IT WRAPS AROUND
                AISC 0F         ;TO THE OTHER END.  DATA IS
                XAD 1,0E        ;NOT DESTROYED BY MOVING
                CLRA            ;CURSOR
                AISC 01
                LBI 1,0F
                XOR
                JMP SKIP
OK:             COMP
SKIP:           LBI 1,0E
                X 0
                RET
CURBAC:         LDD 1,0F        ;MOVES CURSOR BACK ONE
                AISC 01         ;CHARACTER.  DOES NOT
                JMP GOOD        ;DESTROY DATA AS IT IS MOVED
                LBI 1,0E        ;IF MOVED BEYOND THE
                CLRA            ;END OF THE DISPLAY IT
                AISC 01         ;WRAPS AROUND TO THE OTHER
                XOR             ;END
                X 0
                JMP START
GOOD:           XAD 1,0F
                JMP START

SPECIAL:        LDD 1,0C        ;CONTROL CHAR. HAS BEEN
                AISC 03         ;DETECTED
                JMP NOTRET
                JMP RESET       ;RETURN CLEARS DISPLAY,STARTS
                                ;PROGRAM OVER
NOTRET:         AISC 01         ;NOT RETURN, CHECK FOR CURSOR
                JMP CFOR        ;FORWARD
                JMP CURBAC      ;BY DEFAULT, CURSOR BACKWARDS

CFOR:           JSR CURFOR
                JMP START

            ;DISPLAY LOOP
```

```
;THIS IS THE DISPLAY LOOP OF THE PROGRAM. UNLESS A NEW CHARACTER
;HAS BEEN ENTERED AND IS BEING STORED, THE PROGRAM IS ALWAYS IN
;THIS DISPLAY LOOP. IT LOOKS UP THE CHARACTER FONT, SHIFTS THE
;CHARACTER DATA OUT THE SERIAL PORT AND CONTROLS THE GRID DRIVER.

START:      LBI 2,15        ;DISPLAY LOOP POINTER
            JSR HERE        ;GOTO DISPLAY LOOP
            LBI 3,03        ;SECOND DISPLAY LOOP POINTER
            JSR HERE        ;GOTO DISPLAY LOOP
            OGI 09          ;LOADS A 1 IN GRID DRIVER
            OGI 0D
            OGI 09

            JMP START

        ;CHECKS FOR NEW CHAR

HERE:       RC
            ININ
            AISC 15
            JMP OLDCHR
            JMP NEW

        ;DISPLAY LOOP FOR OLD CHAR AND
        ;  LOOK UP

OLDCHR:     LD 2            ;LOOKS UP FIRST BYTE OF CHR.FONT
            JSR DATA4       ;    200H+ASCII VALUE
            AISC 06         ;ADDS 06H TO MSB OF ASCII
            JSR DATA2       ;LOOKS UP SECOND BYTE OF CHR FONT
            AISC 0A         ;ADDS 0AH TO MSB OF ASCII
            JSR DATA2       ;LOOKS UP THIRD BYTE OF CHR. FONT
            JSR DATA3       ;LOOKS UP THIRD BYTE OF CHR. FONT
                            ;    AT 300H+ASCII VALUE
            AISC 06         ;ADDS 06H TO MSB OF ASCII VALUE
            OGI 02          ;TURNS ON BLANKING CONTROL
            JSR DATA3       ;LOOKS UP LAST BYTE OF CHR. FONT
        ;CLOCKS A 0 IN GRID DRIVER

            OGI 0A          ;ENABLE,BLANKING CONTROL
            OGI 0E          ;ENABLE,BLANKING CONTROL,CLOCK
            OGI 0A          ;ENABLE,BLANKING CONTROL
            OGI 00          ;A 0 SHIFTED IN

            LD 0
            XDS 2
            JMP HERE
            RET

RIGHT:      LBI 3,15
            CQMA
            JSR SHIFT       ;OUTPUTS A
            X 0             ;NEW DATA
            JSR SHIFT       ;OUTPUTS A
            LEI 01          ;COUNTER MODE
            LDD 3,14        ;1,0 IN A
            XABR            ;A IN BR
            LDD 3,13        ;1,1 IN A
            CAB             ;A IN BD
            LD 2
            RET
```

Section 3 of COPS Software (Continued)

```
POINTER:        LEI 01          ;COUNTER MODE
                XAS             ;A IN SIO
                XABR            ;BR IN A
                AISC 02         ;ADD 2
                XAD 3,14        ;A IN 1,0
                CBA             ;BD IN A
                XAD 3,13        ;A IN 1,1
                LBI 3,15
                XAS             ;SIO IN A
                LEI 08          ;SERIAL MODE
                JMP RIGHT
          ;SHIFTS OUT SERIAL PORT

SHIFT:          LEI 08          ;THIS ROUTINE SHIFTS THE DATA
                SC              ;FROM THE SI/O REGISTER OUT
                XAS             ;THE SERIAL PORT WITH EACH
                NOP             ;CLOCK CYCLE
                NOP
                RC
                XAS
                RET

          .=0200
DATA3:          LQID
                JMP RIGHT
DATA4:          LQID
                JMP POINTER

          .=0300
DATA3:          LQID
                JMP RIGHT
```

Section 4 of COPS Software

```
;THE CHARACTER FONTS FOR THE CHARACTERS WITH ASCII VALUES BETWEEN
20H AND 5AH HAVE BEEN STORED IN THIS SECTION OF THE PROGRAM.

          ;DATA FOR FIRST 2 BYTES OF EACH
          ;    CHAR.

          .=0220
     .WORD 001, 001, 001, 021, 021, 0C1, 061, 001
     .WORD 031, 001, 041, 011, 001, 011, 001, 001
     .WORD 071, 001, 041, 081, 011, 0E1, 031, 081
     .WORD 061, 061, 001, 001, 001, 021, 001, 041
     .WORD 071, 031, 081, 071, 081, 0F1, 0F1, 071
     .WORD 0F1, 081, 081, 0F1, 0F1, 0F1, 0F1, 071
     .WORD 0F1, 071, 0F1, 061, 081, 0F1, 0F1, 0F1
     .WORD 0C1, 0C1, 081
```

```
      ;DATA FOR SECOND 2 BYTES OF EACH
      ;    CHAR.

      .=0280
.WORD 000, 000, 0C1, 0F9, 0A4, 095, 02D, 000
.WORD 088, 000, 054, 020, 000, 020, 000, 014
.WORD 01D, 082, 003, 005, 058, 045, 0AC, 001
.WORD 02D, 023, 000, 000, 020, 058, 001, 001
.WORD 00D, 0AE, 0F3, 00D, 0F3, 02F, 02F, 00D
.WORD 02E, 003, 00D, 02E, 00E, 08E, 08E, 00D
.WORD 02F, 00D, 02F, 025, 001, 00C, 008, 00C
.WORD 056, 040, 017

      ;THIRD 2 BYTES OF DATA FOR EACH CHAR.


      .=02C0
.WORD 000, 0E3, 000, 0AC, 0FB, 040, 0A5, 083
.WORD 00A, 002, 0F3, 0F1, 034, 040, 008, 040
.WORD 046, 0F7, 02E, 046, 021, 086, 046, 02E
.WORD 046, 046, 0A0, 0B4, 0A0, 0A0, 015, 022
.WORD 0E6, 042, 04E, 006, 00E, 046, 042, 046
.WORD 040, 0F7, 006, 0A0, 004, 080, 0E0, 006
.WORD 042, 026, 062, 046, 0F3, 004, 008, 034
.WORD 040, 070, 046

      ;FOURTH TWO BYTES OF DATA FOR EACH CHAR.


      .=0320
.WORD 000, 008, 007, 0F7, 0AA, 031, 028, 000
.WORD 008, 02A, 049, 080, 000, 080, 000, 001
.WORD 01D, 018, 09C, 09D, 0F7, 01D, 09C, 084
.WORD 09C, 0AC, 000, 000, 022, 041, 041, 08C
.WORD 0DC, 082, 09C, 01C, 01C, 09C, 084, 09C
.WORD 080, 01C, 0EF, 022, 018, 002, 020, 01C
.WORD 084, 02C, 0A4, 09C, 00C, 018, 028, 010
.WORD 041, 009, 01D

      ;LAST BYTES OF DATA FOR EACH CHAR.


      .=0380
.WORD 000, 000, 000, 082, 084, 064, 0A0, 000
.WORD 000, 083, 044, 001, 000, 001, 000, 004
.WORD 0C7, 020, 026, 0CC, 080, 0C9, 0C8, 00E
.WORD 0C6, 087, 000, 000, 028, 082, 001, 006
.WORD 027, 0E3, 0C6, 044, 0C7, 028, 008, 0C5
.WORD 0EF, 028, 008, 028, 020, 0EF, 0EF, 0C7
.WORD 006, 0A7, 026, 0C4, 008, 0CF, 08F, 0CF
.WORD 06C, 00C, 02C


.END
```

MICROWIRE™ Serial Interface

National Semiconductor
Application Note 452
Abdul Aleaf

INTRODUCTION

MICROWIRE is a simple three-wire serial communications interface. Built into COPS™, this standardized protocol handles serial communications between controller and peripheral devices. In this application note are some clarifications of MICROWIRE logical operation and of hardware and software considerations.

LOGICAL OPERATION

The MICROWIRE interface is essentially the serial I/O port on COPS microcontrollers, the SIO register in the shift register mode. SI is the shift register input, the serial input line to the microcontroller. SO is the shift register output, the serial output line to the peripherals. SK is the serial clock; data is clocked into or out of peripheral devices with this clock.

It is important to examine the logical diagram of the SIO and SK circuitry to fully understand the operation of this I/O port *(Figure 1)*.

The output at SK is a function of SYNC, EN0, CARRY, and the XAS instruction. If CARRY had been set and propagated to the SKL latch by the execution of an XAS instruction, SYNC is enabled to SK and can only be overridden by EN0 *(Figure 2)*. Trouble could arise if the user changes the state of EN0 without paying close attention to the state of the latch in the SK circuit.

If the latch is set to a logical high and the SIO register enabled as a binary counter, SK is driven high. From this state, if the SIO register is enabled as a serial shift register, SK will output the SYNC pulse immediately, without any intervening XAS instruction.

The SK clock (SYNC pulse) can be terminated by issuing an XAS instruction with CARRY = 0 *(Figure 3)*.

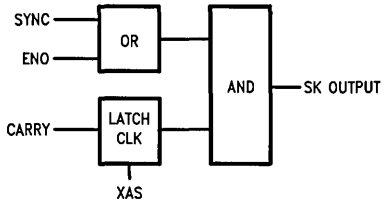


FIGURE 1. Logical Diagram of SK Circuit

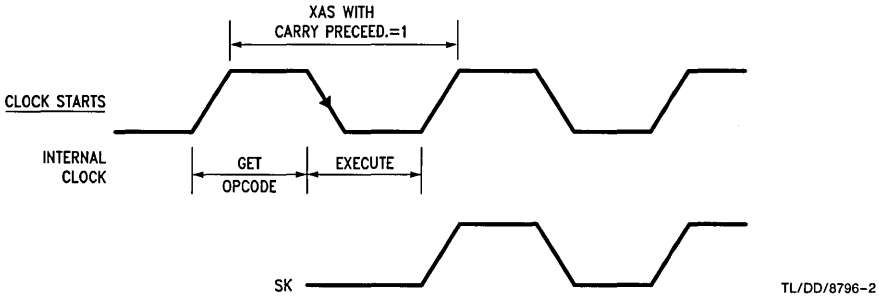


FIGURE 2. SK Clock Starts

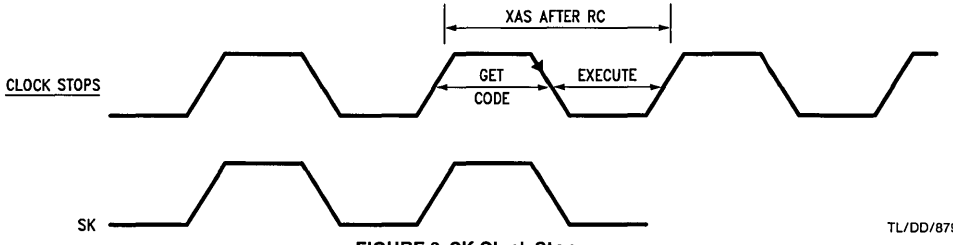


FIGURE 3. SK Clock Stops

The SIO register can be compared to four master-slave flip-flops shown in *Figure 4*. The masters are clocked by the rising edges of the internal clock. The slaves are clocked by the falling edges of the internal clock. Upon execution of an XAS, the outputs of the masters are exchanged with the contents of the accumulator (read and overdrive) in such a way that the new data are present at the inputs of the four slaves when the falling edge of the internal clock occurs. The content of the accumulator is, therefore, latched respectively in the four slave flip-flops and bit 3 appears directly on SO.

This means that:

a) SO will be shifted out upon the falling edges of SK and will be stable during rising edges of SK.

b) SI will be shifted in upon the rising edge of SK, and will be stable when executing, i.e., an XAS instruction.

The shifting function is automatically performed on each of the four instruction cycles that follow an XAS instruction *(Figure 5)*.

When the SIO register is in the shift register mode (EN0 = 0), it left shifts its contents once each instruction cycle. The data present on the SI input is shifted into the least significant bit (bit 0) of the serial shift register. SO will output the most significant bit of the SIO register (bit 3) if EN3 = 1. Otherwise, SO is held low. The SK is a logic controlled clock which issues a pulse each instruction cycle. To ensure that the serial data stream is continuous, an XAS instruction must be executed every fourth time. Serial I/O timing is related to instruction cycle timing in the following way:

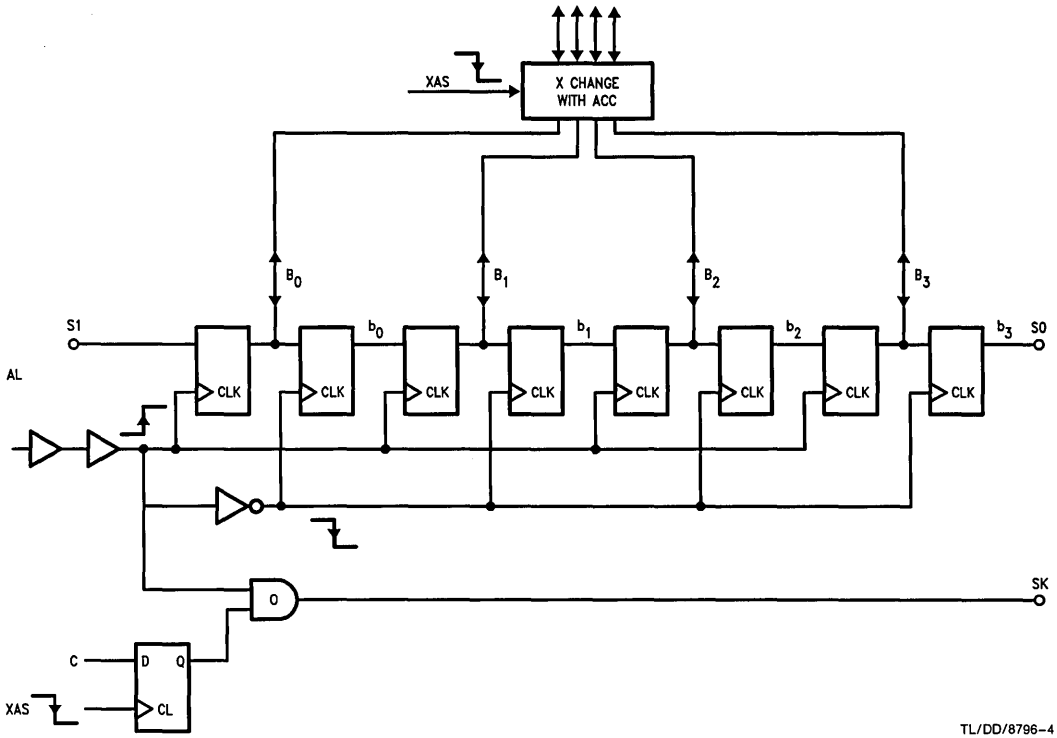


FIGURE 4

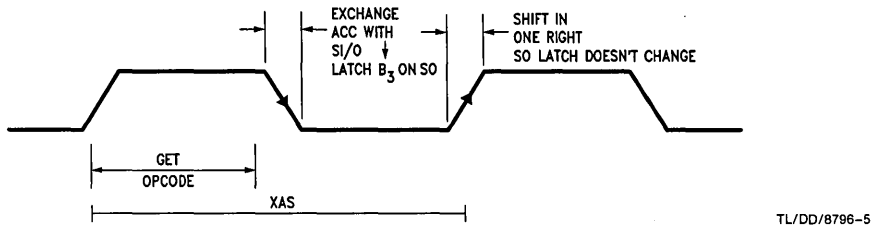
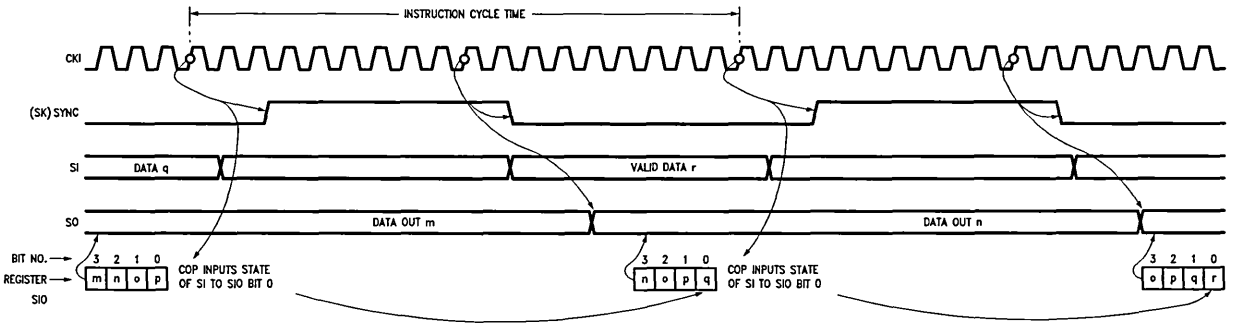
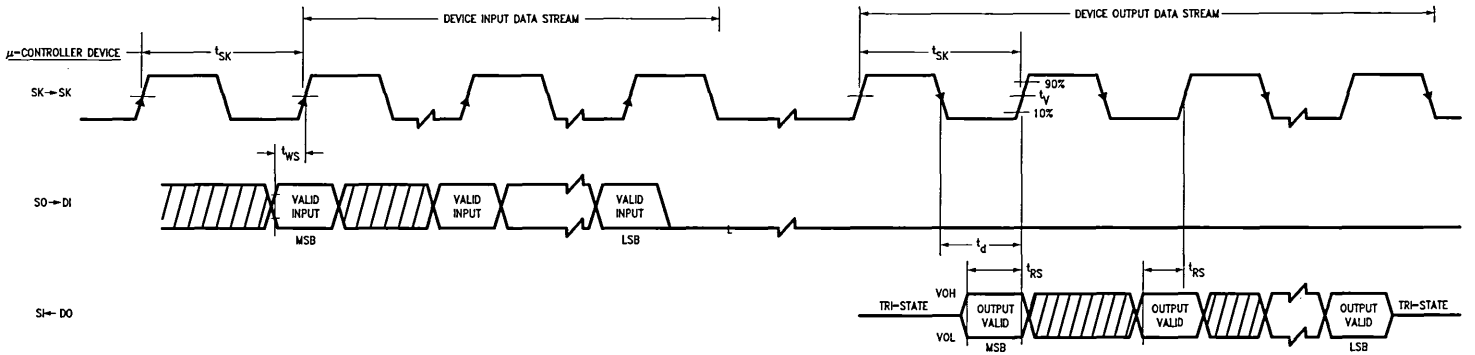


FIGURE 5. XAS Sequence



TL/DD/8796-6

FIGURE 6. Serial I/O Timing



TL/DD/8796-7

FIGURE 7. MICROWIRE Serial Data Exchange Timing

To write to device: $t_{ns} > t_{setup}$

To read from device: $t_d \ll t_{SK} - t_r$; $t_{RS} > t_{SK}/4$

Where: t_{WS} is MICROWIRE write data-in (DI) setup time,

t_{setup} is device data sheet min data setup time to latch in valid data,

t_{SK} is system clock (SK) cycle time (Recommended 50% duty cycle),

t_r is rise time (10% to 70% bout) of system clock (SK),

t_d is device actual delay time before data-out (DO) valid and

t_{RS} is minimum data setup time for controller to shift-in valid data

The first clock rising edge of the instruction cycle triggers the low-to-high transition of SYNC output via SK. At this time, the processor reads the state of SI into SIO bit 0, shifting the current bits 0–2 left. Halfway through the cycle (shown in *Figure 6* as the eight clock rising edge), SK is reset low and the new SIO bit 3 is outputted via SO.

INTERFACING CONSIDERATIONS

To ensure data exchange, two aspects of interfacing have to be considered: 1) serial data exchange timing; 2) fan-out/fan-in requirements. Theoretically, infinite devices can access the same interface and be uniquely enabled sequentially in time. In practice, however, the actual number of devices that can access the same serial interface depends on the following: system data transfer rate, system supply requirement capacitive loading on SK and SO outputs, the fan-in requirements of the logic families or discrete devices to be interfaced.

HARDWARE INTERFACE

Provided an output can switch between a HIGH level and a LOW level, it must do so in a predetermined amount of time for the data transfer to occur. Since the transfer is strictly synchronous, the timing is related to the system clock (SK) *(Figure 7)*. For example, if a COPS controller outputs a value at the falling edge of the clock and is latched in by the peripheral device at the rising edge, then the following relationship has to be satisfied:

$$t_{DELAY} + t_{SETUP} \leq t_{CK}$$

where t_{CK} is the time from data output starts to switch to data being latched into the peripheral chip, t_{SETUP} is the setup time for the peripheral device where the data has to be at a valid level, and t_{DELAY} is the time for the output to read the valid level. t_{CK} is related to the system clock provided by the SK pin of the COPS controller and can be increased by increasing the COPS instruction cycle time.

The maximum t_{SETUP} is specified in the peripheral chip data sheets. The maximum t_{DELAY} allowed may then be derived from the above relationship.

Most of the delay time before the output becomes valid comes from charging the capacitive load connected to the output. Each integrated circuit pin has a maximum load of 7 pF. Other sources come from connecting wires and connection from PC boards. The total capacitive load may then be estimated. The propagation delay values given in data sheets assume particular capacitive loads (e.g. V_{CC} = 5V, V_{OH} = 0.4V , loading = 50 pF, etc.).

If the calculated load is less than the given load, those values should be used. Otherwise, a conservative estimate is to assume that the delay time is proportional to the capacitive load.

If the capacitive load is too large to satisfy the delay time criterion, then three choices are available. An external buffer may be used to drive the large load. The COPS instruction cycle may be slowed down. An external pullup resistor may be added to speed up the LOW level to HIGH level transition. The resistor will also increase the output LOW level and increase the HIGH level to LOW level transition time, but the increased time is negligible as long as the output LOW level changes by less than 0.3V. For a 100 pF load, the standard COPS controller may use a 4.7k external resistor, with the output LOW level increased by less than 0.2V. For the same load, the low power COPS controller may use a 22k resistor, with the SO and SK LOW levels increased by less than 0.1V.

Besides the timing requirements, system supply and fan-out/fan-in requirements also have to be considered when interfacing with MICROWIRE. For the following discussion, we assume single supply push-pull outputs for system clock (SK) and serial output (SO), high-impedance input for serial input (SI).

To drive multi-devices on the same MICROWIRE, the output drivers of the controller need to source and sink the total maximum leakage current of all the inputs connected to it and keep the signal level within the valid logic "1" or logic "0". However, in general, different logic families have different valid "1" and "0" input voltage levels. Thus, if devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Similarly, devices with TRI-STATE® outputs, when connected to the SI input, must satisfy the minimum valid input level of the controller and the maximum TRI-STATE leakage current of all outputs.

So, for devices that have incompatible input levels or source/sink requirements, external pull-up resistors or buffers are necessary to provide level-shifting or driving.

SOFTWARE INTERFACE

The existing MICROWIRE protocol is very flexible, basically divided into two groups:

1) 1AAA.....ADDD.....D

   where leading 1 is the start bit and leading zeroes are ignored.

   AAA.....A is device variable instruction/address word.

   DDD.....D is variable data stream between controller and device.

2) No start bit, just bit stream, i.e., bbb.....b

where b is a variable bit stream. Thus, device has to decode various fields within the bit stream by counting exact bit position.

SERIAL I/O ROUTINES

Routines for handling serial I/O are provided below. The routines are written for 16-bit transmissions, but are trivially expandable up to 64-bit transmissions by merely changing the initial LBI instruction. The routines arbitrarily select register 0 as the I/O register. It is assumed that the external device requires a logic low chip select. It is further assumed that chip select is high and SK and SO are low on entry to the routines. The routines exit with chip select high, SK and SO low. GO is arbitrarily chosen as the chip select for the external device.

SERIAL DATA OUTPUT

This routine outputs the data under the conditions specified above. The transmitted data is preserved in the microcontroller.

```
OUT2:  LBI   0,12    ; point to start of
                       data word
       SC
       OGI   14      ; select the external
                       device
```

TABLE I. MICROWIRE Standard Family

Features		Part Number							
		DS3906	MM545X	COP470	COP472	COP430 (ADC83X)	COP498/499	COP452L	COP494 (NMC9306)
GENERAL									
Chip Function		AM/PM PLL	LED Display Driver	VF Display Driver	LCD Display Driver	A/D	RAM & Timer	Frequency Generator	E^2PROM
Process		ECL	NMOS	PMOS	CMOS	CMOS	CMOS	NMOS	NMOS
V_{CC} Range		4.75V–5.25V	4.5V–11V	−9.5V to −4.5V	3.0V–5.5V	4.5V–0.3V	2.4V–5.5V	4.5V–6.3V	4.5V–5.5V
Pinout		20	40	20	20	8/14/20	14/8	14	14
HARDWARE INTERFACE									
Min V_{IH}/Max V_{IL}		2.1V/0.7V	2.2V/0.8V	−1.5V/−4.0V	$0.7V_{CC}$/0.8V	2.0V/0.8V	$0.8V_{CC}$/$0.4V_{CC}$	2.0V/0.8V	2.0V/0.8V
SK Clock Range		0–625 kHz	0–500 kHz	0–250 kHz	4–250 kHz	10–200 kHz	4–250 kHz	25–250 kHz	0–250 kHz
Write Data DI	Setup Min	0.3 μs	0.3 μs	1.0 μs	1 μs	0.2 μs	0.4 μs	800 ns	0.4 μs
	Hold Min	0.8 μs	(3)	50 ns	100 ns (Note 1)	0.2 μs	0.4 μs	1.0 μs	0.4 μs
Read Data Prop Delay		(Note 4)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	2 μs (Note 2)	1 μs (Note 2)	2.0 μs
Chip Enable	Setup	0.3 μs	0.4 μs	1.0 μs Min	1 μs (Note 1)	0.2 μs	0.2 μs (Note 1)	(Note 3)	0.2 μs
	HOLD	0.8 μs	(Note 3)	1.0 μs Min	1 μs (Note 2)	0.2 μs	0 (Note 2)	(Note 3)	0
Max Frequency Range	AM	8 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
	FM	120 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
Max Osc Freq.		(Note 3)	(Note 3)	250 kHz	(Note 3)	(Note 3)	2.1 MHz (−21) 32 kHz (−15)	256–2100 kHz (−4) 64–525 kHz (−2)	(Note 3)
SOFT									
Serial I/O Protocol		11D1...D20	1D1...D35	8 Bits At a Time	b1...b40	1xxx	1yyxxD6...D0 Start Bit	1yxxxx	1AA...DD
Instruction/ Address Word		None	None	None	None	(Note 4)	(Note 4)	(Note 4)	(Note 4)

Note 1: Reference to SK rising edge.

Note 2: Reference to SK falling edge.

Note 3: Not defined.

Note 4: See data sheet for different modes of operation.

```
        LEI  8      ; enable shift
                      register mode
        JP   SEND2
SEND1:  XAS
SEND2:  LD          ; data output loop
        XIS
        JP   SEND1
        XAS         ; send last data
        RC
        CLRA
        NOP
        XAS         ; turn SK clock off
        OGI  15     ; deselect the device
        LEI  0      ; turn SO low
        RET
```

The code for reading serial data is almost the same as the serial output code. This should be expected because of the nature of the SIO register and the XAS instruction.

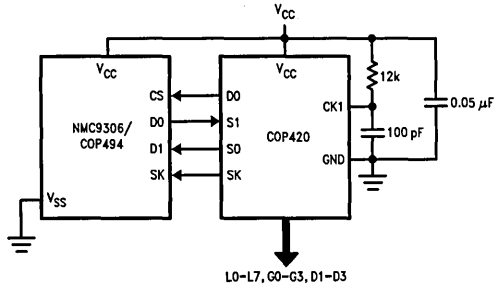
MICROWIRE STANDARD FAMILY

A whole family of off-the-shelf devices exists that is directly compatible with MICROWIRE serial data exchange standard. This allows direct interface with the COPS family of microcontrollers.

Table I provides a summary of the existing devices and their functions and specifications.

TYPICAL APPLICATION

Figure 8 shows pin connection involved in interfacing an NMC9306/COP494 E²PROM with the COP420 microcontroller.

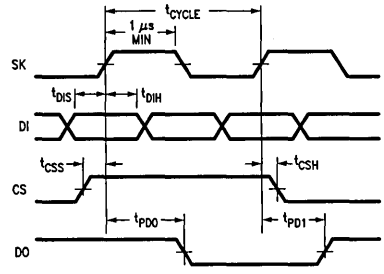


TL/DD/8796-8

FIGURE 8. NMC9306/COP494-COP420 Interface

The following points have to be considered:

1. For COP494 the SK clock frequency should be in the 0 kHz–250 kHz range. This is easily achieved with COP420 running at 4 µs–10 µs instruction cycle time (SK period is the COP420 instruction cycle time). Since the minimum SK clock high time is greater than 1 µs, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max.
2. CS low period following an E/W instruction must not exceed 30 ms maximum. It should be set at typical or minimum spec of 10 ms. This is easily done in software using the SKT timer on COP420.



TL/DD/8796-9

FIGURE 9. NMC9306/COP494 Timing

3. As shown in WRITE timing diagram, the start bit on DI must be set by a "0" to "1" transition following a CS enable ("0" to "1") when executing any instruction. One CS enable transition can only execute one instruction.
4. In the read mode, following an instruction and data train, the DI can be a "don't care," while the data is being outputted, i.e., for the next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
5. The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI until another CS LO to HI transition starts a new instruction cycle.
6. After a read cycle, the CS must be brought low for one SK clock cycle before another instruction cycle starts.

INSTRUCTION SET

Commands	Opcode	Comments
READ	10000A3A2A1A0	Read Register 0–15
WRITE	11000A3A2A1A0	Write Register 0–15
ERASE	10100A3A2A1A0	Erase Register 0–15
EWEN	111000 0 0 1	Write/Erase Enable
ENDS	111000 0 1 0	Write/Erase Disable
***WRAL	111000 1 0 0	Write All Registers
ERAL	111000 1 0 1	Erase All Registers

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ— After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE— Write command shifted in followed by data in (16 bits) then CS pulsed low for 10 ms minimum.

ERASE

ERASE ALL—Command shifted in followed by CS low.

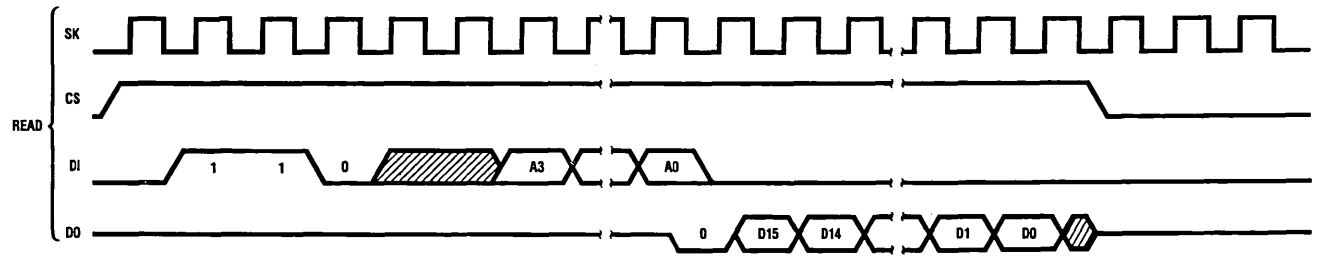
WRITE ALL—Pulsing CS low for 10 ms.

WRITE

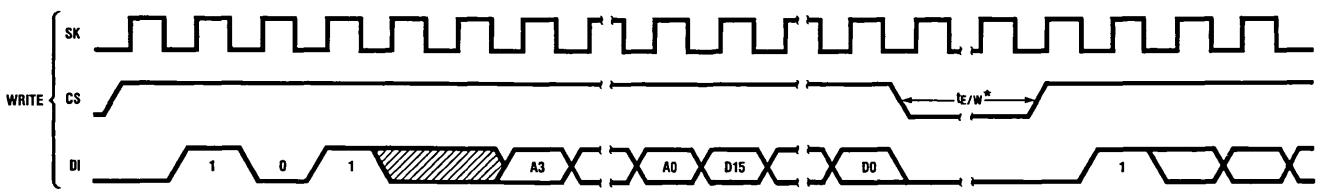
ENABLE/DISABLE—Command shifted in.

***(This instruction is not speced on Data sheet.)

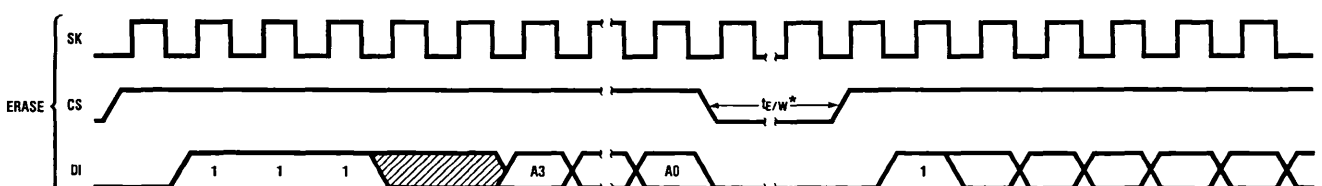
Instruction Timing



TL/DD/8796–10



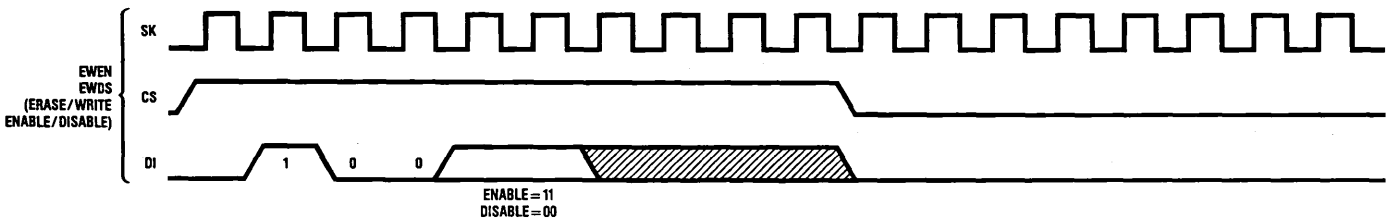
TL/DD/8796–11



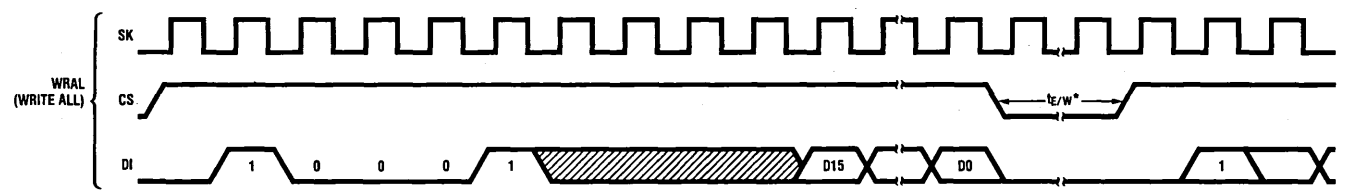
TL/DD/8796–12

*$t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

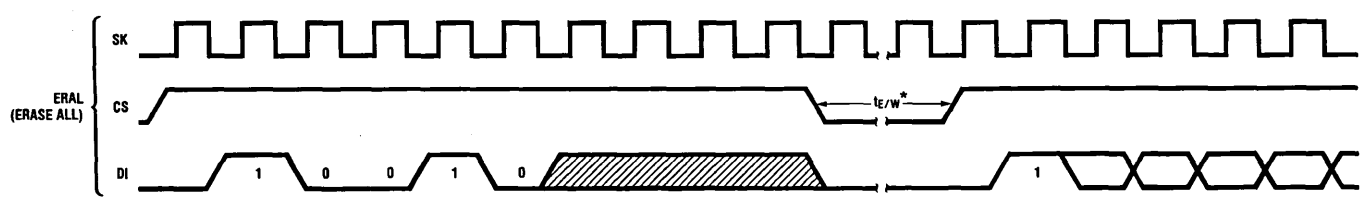
TIMING WAVEFORMS (Continued)



TL/DD/8796–13



TL/DD/8796–14



TL/DD/8796–15

*$t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

I/O ROUTINE TO EVALUATE COP494

```
 1                          .TITLE      E494,  "I/O ROUTINE TO EVALUATE COP494"
 2     01A4                 .CHIP       420
 3     0000                 .PAGE       0
 4
 5                  ;THIS IS I/O ROUTINE TO EVALUATE COP494
 6                  ;
 7                  ;
 8
 9                  ;RAM VARIABLES DECLARATIONS:
10     000E                 COMMAND = 0, 14          ;494 8BITS INST/ADDR WORD
11     001C                 RWDATA  = 1, 12          ;494 16BITS R/W DATA BUFFER
12
13 000 00          PON:     CLRA                     ;POWER-ON INIT
14 001 32                   RC                       ;RESET SK CLOCK
15 002 4F                   XAS
16 003 3F          CLRAM:   LBI     3, 0            ;CLEAR RAM FROM 7, 0 TO 0, 15
17 004 00          CLR:     CLRA                     ;
18 005 04                   XIS                      ;
19 006 C4                   JP      CLR              ;CONTI CLEAR REG
20 007 12                   XABR                     ;(A) TO BR
21 008 5F                   AISC    15               ;REG 0 CLEARED?
22 009 600F        DONE:    JMP     C494DR           ;Y, DONE CLEAR RAM, CALL 494 D
23 00B 12                   XABR                     ;N, DEC BR
24 00C C4                   JP      CLR              ;CONTI CLEAR REG TILL DONE
25 00D 44                   NOP
26 00E 44                   NOP
27
28                 ;***     START 494 DRIVER SAMPLE CALLING SEQUENCE   ***
29
30                 C494DR:                          ;INIT CALLING SEQUENCE
31 00F 3350                 OGI     0                ;GO=L TO DESELECT 494
32 011 3368                 LEI     8                ;ENABLE SIO AS S.R.
33                 ERASE:
34 013 0D                   LBI     COMMAND          ;PRELOAD 494 ERASE REG A3-A0
35 014 7C                   STII    0C               ;PRELOAD 494 ERASE INST
36 015 70                   STII    0                ;SELECT REG A3-A0
37 016 690E                 JSR     WI4P4            ;SEND IT
38                 WEEN:
39 01B 0D                   LBI     COMMAND          ;LOAD 494 WHEN REG A3-A0
40 019 73                   STII    3                ;PRELOAD 494 WREN INST
41 01A 70                   STII    0                ;SELECT REG A3-A0
42 01B 690E                 JSR     WI494            ;SEND IT
43                 WRITE:
44 01D 0D                   LBI     COMMAND          ;PRELOAD WR REG A3-A0
45 01E 74                   STII    4                ;PRELOAD 494 WRITE INST
46 01F 70                   STII    0                ;SELECT REG A3-A0
47 020 1B                   LBI     RWDATA           ;PRELOAD 494 SAMPLE WRITE DATA
48 021 75                   STII    5
49 022 7A                   STII    0N
50 023 75                   STII    5
```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```
 51 024 7A             STII    0A
 52 025 6900           JSR     WD494           ;SEND THEM TO 494
 53           READ:
 54 027 0D             LBI     COMMAND         ;PRELOAD READ REG A3-A0
 55 028 78             STII    8               ;PRELOAD 494 READ INST
 56 029 70             STII    0               ;SELECT REG A3-A0
 57 02A 6908           JSR     RD494           ;READ 494 DATA BACK VIA SI
 58 02C 44             NOP
 59 02D 44             NOP
 60
 61     0080           .PAGE   2               ;SUBROUTINE PAGE
 62 080 32    SETUP:   RC                      ;RESET SK BEFORE SELECT 494
 63 081 4F             XAS
 64 082 3351           OGI     1               ;GO=1 TO SELECT 494
 65 084 00             CLRA                    ;ENSURE SO=L BEFORE GEN START B
 66 085 22             SC                      ;
 67 086 4F             XAS                     ;TURN ON SK CLOCK
 68 087 00             CLRA                    ;GENERATE 494 START BIT
 69 088 51             AISC    1               ;
 70 089 22             SC                      ;
 71 08A 4F             XAS                     ;SEND IT AS MSB VIA SO
 72 08B 0D             LBI     COMMAND         ;FETCH 1ST INST/ADDR WORD
 73 08C 05             LD
 74 08D 44             NOP                     ;
 75 08E 4F             XAS                     ;SEND IT (MSB OF INST FIRST)
 76 08F 0E             LBI     COMMAND+1       ;FETCH 2ND INST/ADDR NIBBLE
 77 090 05             LD
 78 091 44             NOP
 79 092 4F             XAS                     ;SEND IT
 80 093 1B             LBI     RWDATA          ;POINT TO READ/WRITE DATA BUFFER
 81 094 48             RET                     ;RET OF SETUP
 82
 83 095 00    TWEDLY:  CLRA                    ;VPP WIDTH, TWE>20MS @ 4Us/INST
 84 096 5B    TWECONT: AISC    11              ;5 SKT LOOPS?
 85 097 99             JP      . + 2           ;N,CONTI
 86 098 48    TWEDONE: RET                     ;Y,DONE
 87 099 41             SKT                     ;
 88 09A 99             JP      . -1            ;
 89 09B 96             JP      TWECONT         ;CONTI TWE TIME
 90
 91 09C 48    RET;     RET                     ;2 CYCLES DELAY
 92
 93     0100           .PAGE   4               ;
 94
 95           ;***     START 494 I/O DRIVER SUBROUTINE    ***
 96
 97 100 80    WD494:   JSRP    SETUP           ;ENTRY TO WRITE 494 REG A3-A0
 98 101 05    RWLOOP:  LD                      ;R/W 494 16 DATA BITS
 99 102 4F             XAS                     ;
100 103 04             XIS                     ;
```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```
101 104 C1          JP    RWLOOP
102 105 3350        OGI   0          ;DESELECT 494 AFTER R/W DATA
103 107 D1          JP    FINI       ;
104 108 80    RD494: JSRP  SETUP      ;ENTRY TO RD 494 REG A3-A0
105 109 00          CLRA             ;FINISH SEND OUT A3-A0 VIA SO
106 10A 44          NOP              ;
107 10B 44          NOP              ;WAIT 1BIT TIME FOR VALID D15
108 10C 44          NOP
109 10D C1          JP    RWLOOP       ;
110 10E 80    WI494: JSRP  SETUP      ;ENTRY TO WRITE INST TO 494
111 10F 00          CLRA             ;ENSURE SO = L
112 110 4F          XAS              ;
113 111 00    FINI: CLRA             ;ENSURE SO = L BETWEEN INST
114 112 3350        OGI   0          ;DESELECT 494 BETWEEN INST WRIT
115 114 32          RC               ;
116 115 4F          XAS              ;TURN OFF SK CLOCK
117 116 95          JSRP  TWEDLY     ;DELAY TWE >20MS TO PULSE VPP=21
118 117 48          RET              ;RET OF WD494 OR RD494 OR WI494
119
120                 .END
```

SOFTWARE DEBUG OF SERIAL REGISTER FUNCTIONS

In order to understand the method of software debug when dealing with the SIO register, one must first become familiar with the method in which the COPS MOLE™ (Development System) BREAKPOINT and TRACE operations are carried out. Once these operations are explained, the difficulties which could arise when interrogating the status of the SIO register should become apparent.

SERIAL OUT DURING BREAKPOINT

When the MOLE BREAKPOINTs, the COPS user program execution is stopped and execution of a monitor-type program, within the COP device, is started. At no time does the COP part "idle." The monitor program loads the development system with the information contained in the COP registers.

Note also that single-step is simply a BREAKPOINT on every instruction.

If the COP chip is BREAKPOINTed while a serial function is in progress, the contents of the SIO register will be destroyed.

By the time the monitor program dumps the SIO register to the MOLE, the contents of the SIO register will have been written over by clocking in SI. To inspect the SIO register using BREAKPOINT, an XAS must be executed prior to BREAKPOINT; therefore, the SIO register will be saved in the accumulator.

An even more severe consequence is that the monitor program executes an XAS instruction to get the contents of the SIO register to the MOLE. Therefore, the SK latch is dependent on the state of the CARRY prior to the BREAKPOINT. In order to guarantee the integrity of the SIO register, one must carefully choose the position of the BREAKPOINT address.

As can be seen, it is impossible to single-step or BREAKPOINT through a serial operation in the SIO register.

SERIAL OUT DURING TRACE

In the TRACE mode, the user's program execution is never stopped. This mode is a real-time description of the program counter and the external event lines; therefore, the four external event lines can be used as logic analyzers to monitor the state of any input or output on the COPS device. The external event lines must be tied to the I/O which is to be monitored.

The state of these I/O (external event lines) is displayed along with the TRACE information. The safest way to monitor the real-time state of SO is to use the TRACE function in conjunction with the external event lines.

CONCLUSIONS

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and may permit the COPS controller to be packaged in a smaller package.

COPS™ Based Automobile Instrument Cluster

National Semiconductor
Application Note 453
Venkata T. Gobburu

ABSTRACT

Dedicated microprocessor systems find increasing applications in automobile instrumentation. Fuel injection systems, digital radio tuners and similar applications employing the microcontroller have become common place. This paper describes a cost effective microcontroller implementation of an automobile instrument cluster by the COPS group of National Semiconductor, Santa Clara. The instrument cluster provides a vacuum fluorescent display of the vehicle speed, engine RPM, odometers, battery voltage, engine oil pressure and the fuel level. A modular design involving a single microcontroller in conjunction with peripherals to aid in data acquisition from the transducers allows the quantities to be computed with high accuracies and displayed on a real time basis. The single microcontroller environment places severe restrictions on the availability of RAM and ROM. Coupled with the requirement of real time operation the application poses a non trivial challenge. A nonvolatile RAM accumulates the mileage covered. Hamming code techniques ensure the integrity of the data contained in the nonvolatile memory. Inclusion of diagnostics allows a rapid and thorough check against improper operation of the microcontroller, peripherals and the nonvolatile memory. This paper describes the implementation with a COP444L containing 128 nybbles of RAM and 2K bytes of ROM. A display updation rate of 16 Hz can be comfortably realized.

Over the microcomputer usage has diversified dramatically in its scope and breadth. Dedicated microprocessor systems find increasing application in automobile instrumentation and control. From its inception the automobile has acquired considerable sophistication. Increasing demands have been made of the car. Fuel efficiency, higher acceleration rates, simplicity of control and improved ride quality rank high in the demands made of the car. In response the automobile engine has evolved into a complex machine. Crude methods to control or monitor its performance no longer suffice. Microprocessor based fuel injection techniques and ignition control are becoming quite ubiquitous.

The automobile instrument cluster monitors the engine and regularly updates a status display for the operator's benefit. Pertinent information includes the vehicle speed, the engine crankshaft rotational speed, oil pressure in the engine cylinders, condition of the battery and the mileage accumulated. The instrument cluster provides a visual feedback link to the operator allowing corrective action to be initiated as the need arises.

THE AUTOMOBILE INSTRUMENT CLUSTER

The heart of the Automobile Instrument Cluster (AIC) lies in obtaining raw data from various transducers and manipulating it to a form suitable for feedback to the human operator. The feedback, normally visual, conveys the vehicle speed, the engine rpm, the engine temperature, oil pressure, the battery voltage and the odometer values. The AIC can be viewed as a collection of either inherently independent or weakly linked subtasks. Each subtask can be further partitioned into three blocks viz. of raw data collection, processing and displaying it. The component subtasks, in spite of their high degree of independence, can be grouped on the basis of signal available from the transducers. Grouping the subtasks modularizes the design. Partitioning the design in this manner highlights two groups, the first requires a frequency to be measured and the second a voltage level. The two major groupings are briefly examined.

Transducers for the vehicle speed monitor the driveshaft rotation. Computing the engine rpm involves measuring the crankshaft revolution rate. The two independent problems can be seen to basically consist of measuring revolution rates. Transducers based on Hall effect phenomena have been used with commendable success. Alternately the fact that mounting magnets around the driveshaft circumference generates a known number of pulses per shaft rotation can be used effectively. A normally open cam operated reed switch with closure to ground creates a simple revolution transducer. In all the cases the transducer generates a frequency proportional to the quantity under consideration. Obviously some signal conditioning is required before using the frequency with digital components. The describing function can be simply stated as

$$V = k \times f \quad (1)$$

where

V is the quantity under measurement, the vehicle speed or the engine rotational speed
k is a proportionality constant
f is the transducer freqeuncy output

The proportionality constant, k, can be suitably modified to include changes back and forth between British and metric units.

The problem of measuring the transducer output frequency can be restated to be one of measuring the time period. In case of digital frequencies the equation (1) can be rewritten as

$$V = k/(Ton + Toff) \quad (2)$$

where

Ton is the ON time and
Toff is the OFF time

while the remaining symbols retain their definition from the earlier equation.

The remaining quantities such as the engine temperature, oil pressure, battery voltage and available fuel prove to be slow changing ones. The lower dynamics allow them to be transduced as voltage level signals. Equation (3) states the underlying relation and closely resembles the equations stated above.

$$P = k \times v \quad (3)$$

where

v is the voltage output of the transducer
P is the quantity under measurement
k is the proportionality constant

Evaluating the accumulating mileage depends indirectly upon the vehicle speed subtask. Integrating the signal from the vehicle speed transducer over time allows the mileage to be accumulated. The associated problems of storing the odometer information and ensuring its integrity require error correcting techniques. They are covered in a later section of the paper.

SYSTEM DESCRIPTION

The COPS Group of National Semiconductor, Santa Clara, offers a wide array of microcontrollers and peripherals to suit this application. Judicious selection of peripherals to aid the microcontroller can reinforce the partitioning suggested earlier to considerably simplify the implementation. *Figure 1* presents a functional block diagram of the AIC.

A COP444L four bit microcontroller provides the necessary computing and decision making capability. Equipped with 128 nybbles of RAM space organized in a matrix fashion and 2K ROM space for storage of the control program, the COP444L operating at an instruction cycle rate of 16 microseconds sequentially obtains information from the peripherals and formats the manipulated results to be manageable by the display drivers. Transducers for the vehicle speed and the engine speed provide proportional frequency signals. Two COP452 peripherals, placed in a Waveform Measure Mode, track the ON time and OFF time of the conditioned transducer outputs. Voltage level signals available from the transducers for the engine temperature, oil pressure, battery condition and the fuel tank can be monitored by a COP438, an eight channel A/D converter. An electronically erasable non volatile RAM, the COP494, allows the odometer information to be stored safely under power down conditions.

A combination of LEDs, vacuum fluorescent displays and high intensity lamps comprise the optical elements of the AIC Standard eight segment alphanumeric and bargraph format displays have been used. A 32 segment LED bargraph, controlled by a MM5450 static display driver, displays the engine rpm. Eight segment alphanumeric vacuum fluorescent displays are used for the vehicle speed and the odometer values. Sixteen segment vacuum fluorescent bargraph displays are used for the engine temperature and available fuel quantity. The battery voltage and oil pressure utilize eight segment vacuum fluorescent bargraph displays. Any potentially dangerous situations detected by the COP444L are underlined by high intensity lamps. Five COP470 display drivers multiplex the various displays under the microcontroller's orchestration.

Single pole single throw switches allow the user to select between the British or the metric units, the trip or the accumulated odometer and reset the trip odometer.

SYSTEM DIAGNOSTICS

Diagnostics aid in isolating faulty components within a system. The algorithmic nature of the diagnostic procedure allows it to be implemented via a microprocessor. A great deal of attention has been focused on diagnostics as considerable cost savings can accrue from a microprocessor based scheme minimizing human involvement. Programming the AIC, in addition to its normal functions, with self test capabilities increases its potential for high volume applications. Normally diagnostics imply using independent means to evaluate the system's performance. Attempting to incorporate self test capabilities necessitates adopting an "inside out" strategy. A basic kernel is first evaluated as functioning correctly. Over iterations the kernel expands by establishing correct operation of other modules.

The AIC implementation described in this paper has an extensive repertoire of diagnostics to check the microcontroller and ensure correct operation of the peripherals. The probability of the microcontroller ROM failing proves to be negligibly small compared to a fault developing in the hardware interconnections. Also the idea of encoding in ROM the algorithm to check ROM data proves suspect. Control program stored in the ROM forms the kernel assumed to be functioning correctly. Writing and reading back an alternating pattern of ones and zeros in the microcontroller RAM checks for leakage of data into adjacent locations. Applying a known voltage, derived locally, to one of the four unused channels on the A/D converter allows it to be tested. The architecture of the COP452 peripherals consists of two independent register-counter pairs. The counters count down from the initial value. To test the COP452 both the register counter pairs have to be checked. By placing the two in a Duty Cycle Mode, the counters can be loaded with initial values from the registers and set to count down. The contents of the counters after a predetermined delay can detect incorrect operation of the device. A fault at the level of a register-counter pair can thus be isolated.

The COP494 stores the odometer information. It becomes vital to maintain the integrity of the information stored in the nonvolatile memory. Continuous use of particular locations in the COP494 can result in failures, typically bit dropouts. It is imperative to be capable of recovering from such errors. Requiring a single COP494 unit to last at least the expected lifetime of the vehicle influences the design of the storage scheme. The AIC implementation described in this paper depends upon Hamming encoding techniques to provide single bit error recovery. Subsequent to recovering from a single bit error all data transactions are carried out from a new location. A flashing display sequence alerts the operator of the occurrence of a non-recoverable error. Suspending all normal functions during such conditions can be used to force the vehicle to be taken to an authorized dealer. Breaking up the odometer data into sections allows updating of particular sections as opposed to restoring the whole every time. Such a strategy maximizes the lifetime of the nonvolatile memory.

SOFTWARE DESCRIPTION

The functional objectives of the AIC and the hardware required to realize them have been detailed in earlier sections of the paper. A summary of the software features completes the description and aids in developing a global understanding of the AIC. The AIC software, written in COP microcontroller assembly language, reflects the modular nature of the problem. The finite amount of memory of ROM space available on the COP444L coupled with real time operation requirements makes programming the AIC a non-trivial problem. Each subtask grouping has been organized as a distinct block of code. The microcontroller sequentially processes each subtask. A brief examination of the salient features follows.

It must be borne in mind that the COP452 peripheral captures an instantaneous picture of the frequency. The strength of the magnets, mounted circumferentially on the driveshaft to transduce revolution rate, cannot be precisely controlled. As a result the transducer, although generating a fixed number of pulses per revolution of the driveshaft, produces a pulse train showing both pulse period and duty cycle variations. Directly using the pulse period from the

COP452 leads to erroneous values of the vehicle speed. The computed vehicle speed, under steady vehicle speed conditions, shows excursions on either side of the nominal value. The first AIC implementation studied the application of an essentially single pole filter with different damping constants to exclude the oscillations. Although a sufficiently damped filter can effectively reduce the oscillations the scheme was discarded in lieu of the resulting degradation in response time. The solution lies in basing the vehicle speed computation on pulse period measurements averaged over consecutive pulses. Since the number of pulses per revolution is known, eight in this case, averaging the pulse period over this number minimizes the steady state error and responds fast. The nature of the solution affects the software organization. It falls upon the microcontroller to sample the conditioned output of the transducer and obtain pulse periods for eight consecutive pulses. To achieve this the software adopts a foreground-background organization. Monitoring the transducer output to catch the consecutive pulses forms the background job. The normal functions of the AIC form the foreground job. Additionally a minimal sampling rate has to be maintained to ensure that even at highest attainable vehicle speeds the microcontroller measures consecutive pulses.

The AIC electronically stores the odometer information in the non-volatile memory. Loss of odometer integrity can be disastrous. Consequently the ability to recover from errors in the non-volatile memory becomes very important. The AIC depends on single bit error correcting Hamming coding methods to avoid loss of information. The algorithm processes the odometer nybble fashion and simplifies the related problems of encoding the data prior to storing it and decoding the composite for data retrieval to trivial table lookups. LQID, a powerful member of the microcontroller instruction set, allows an eight bit value to be looked up based on the key value in the addresed RAM location. To minimize ROM space both the encoding and the decoding sections of the algorithm share the same error table and code for table lookups.

The remaining sections of the AIC software, also exhibit a block structure, do not prove to be as subtle. The straight forward code includes routines such as multiplications and divisions to help in the computations and routines allowing the microcontroller to communicate serially over the MICROWIRE™ with the peripherals.

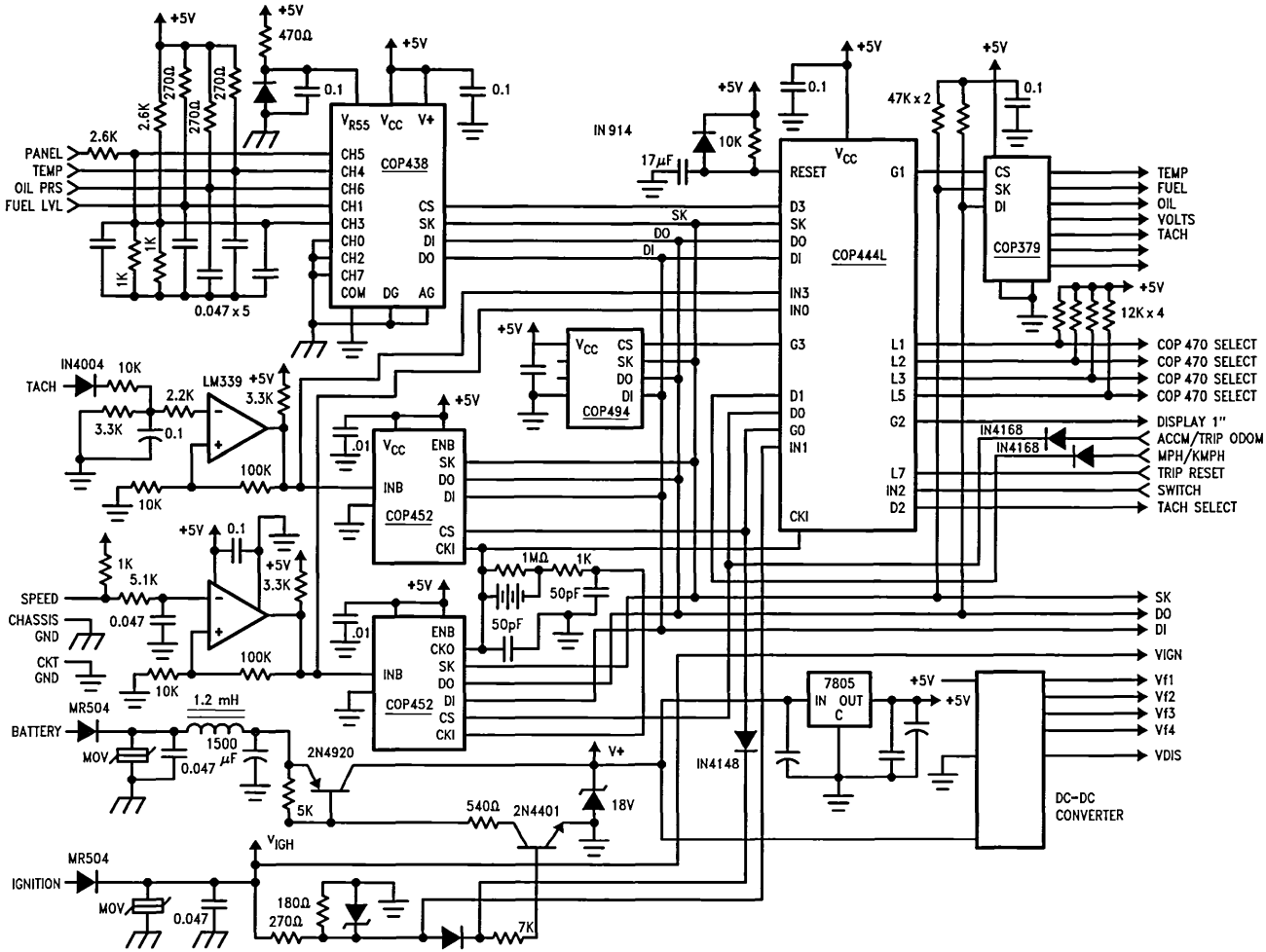
RESULTS AND CONCLUSIONS

The AIC implemented via the COP444L approximately uses 2K of ROM space. The COP444L, running at an instruction cycle time of 16 microseconds, sequences through all the functions in 228 milliseconds. The resulting display updation rate of approximately 4 Hz can be trivially increased to 16 Hz by replacing the COP444L with the equivalently packaged COP440. Table I presents in tabular form the accuracies and speeds at which the different measurements are done. It also shows the proportional speed increases obtainable.

The minimal number of peripherals used combined with the inclusion of diagnostics and error correction emphasize its low cost capabilities. The results serve to validate the feasibility of a cost effective microcontroller based Automobile Instrument Cluster.

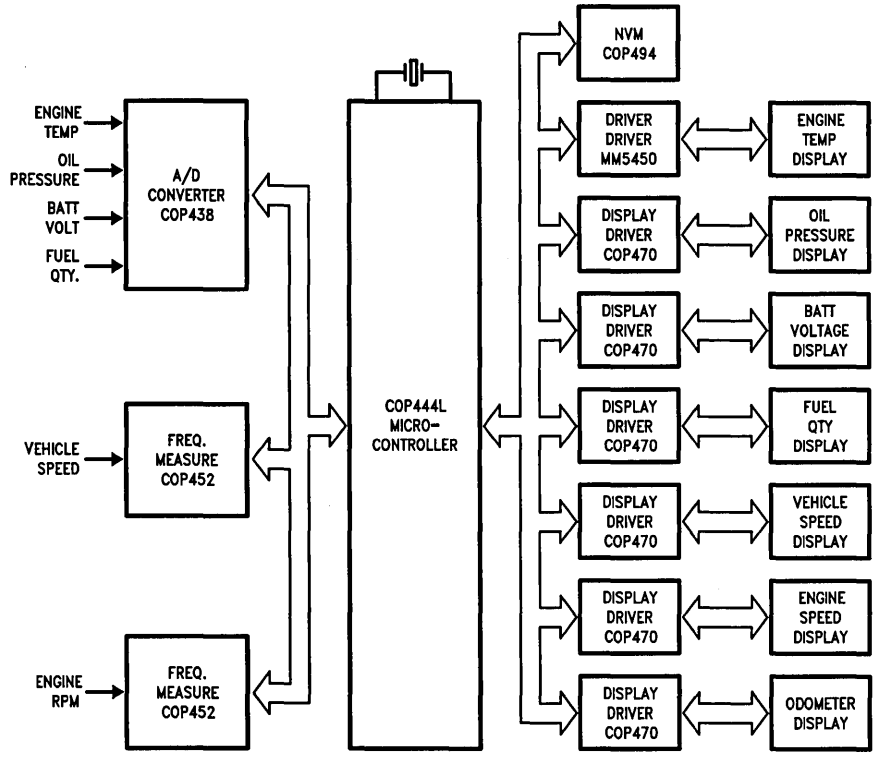
TABLE I. Comparison of Speed and Resolution of Measurements Taken with the COP444L and the COP440

	Measurements with a COP444L		Measurements with a COP440	
	Time Taken μsecs	Resolution Bits	Time Taken μsecs	Resolution Bits
1. Engine rpm	768	17	192	17
2. Vehicle Speed	768	17	192	17
3. Engine Temperature	256	8	64	8
4. Oil Pressure	256	8	64	8
5. Battery Voltage	256	8	64	8
6. Fuel Quantity	256	8	64	8



TL/DD/8798–1

FIGURE 1. Functional Block Diagrams of the AIC



TL/DD/8798–2

FIGURE 2

Automotive Multiplex Wiring

National Semiconductor
Application Note 454
Abdul H. Aleaf

INTRODUCTION

The evolutionary development of vehicle electronic systems has rapidly increased the number of individual wires in the vehicle. The conventional wiring harness will not provide solutions to the problems such as reducing size and weight in addition to meeting cost and reliability objectives. Several approaches have been taken to provide long term solutions. None has succeeded. Miniaturization of cables and wires is one example of a temporary solution.

Multiplexing on the other hand has been regarded as a technique which allows considerable savings to be made in the size and cost of the harness. It can also enhance reliability by reducing the number of electrical connections.

In a multiplex system the control functions will be distributed around the vehicle and complex interconnections between diagnostic terminals, sensors, instruments and switches will not add to the harness complexity. With all its advantages it has not been implemented on a production car yet. The reason has been economical feasibility and lack of suitable semiconductor components for power switching. But, with the rapid technology advances in power FETs and introduction of low cost microcomputers, multiplex wiring can be regarded as a logical successor to conventional wiring systems. Extended development efforts are necessary to introduce a reliable system at reasonable cost.

The Microcontroller Applications Group at National Semiconductor has taken a step towards this goal. A low end multiplex wiring system focusing on asynchronous serial communication in a multi node network has been developed. This paper describes the development of this system on an abstract model which forms the basis for analysis of communication protocol and various node functions.

SYSTEM CONFIGURATION

Figure 1 presents a general view of the system. The system is a centralized single master multiple slave-node scheme. All units are connected together by a balanced twisted pair. The expandable interconnection of different subsystems is achieved with 9600 Baud communication over a standard UART bus. The bus handles the interface between a master controller and the intelligent nodes.

The approach to have a centralized control system offers several advantages as compared against a non-centralized system. It prevents the problem of bus monopolization by a faulty node and is potentially cheaper due to the need for only one complex node (master). The master-slave architecture also prevents bus contention problems.

The master is a COP420L. The COP420L is a 4-bit microcontroller with a software UART that handles asynchronous communication with other processors at speeds up to 9600 Baud.

The use of 4-bit 49¢ microcontrollers (COP413L) at the nodes not only provides intelligence which reduces the required bus bandwidth, it also reduces the incremental cost associated with automotive multiplexing. All standard nodes are identical. One standard program is used. This uniformity contributes to the system flexibility and expandability. External standard nodes may be added to the system to control additional functions. Node types and addresses are selected via external wire jumpers or switches. The slave nodes consist of four remote units to handle functions such as headlamps, tail lamps, etc. These nodes are the front right, front left, rear right and rear left nodes. Incorporated into the system are also a keyboard node, a EIC node and a display node.

The keyboard node may call for a control action at any time. This node is being continuously monitored by the master controller which receives status and processes the command or information.

Overall system intelligence and flexibility is increased by dedicating a node to NS455 the Terminal Management Processor. This node takes the responsibility to display information on a 4" flat CRT display.

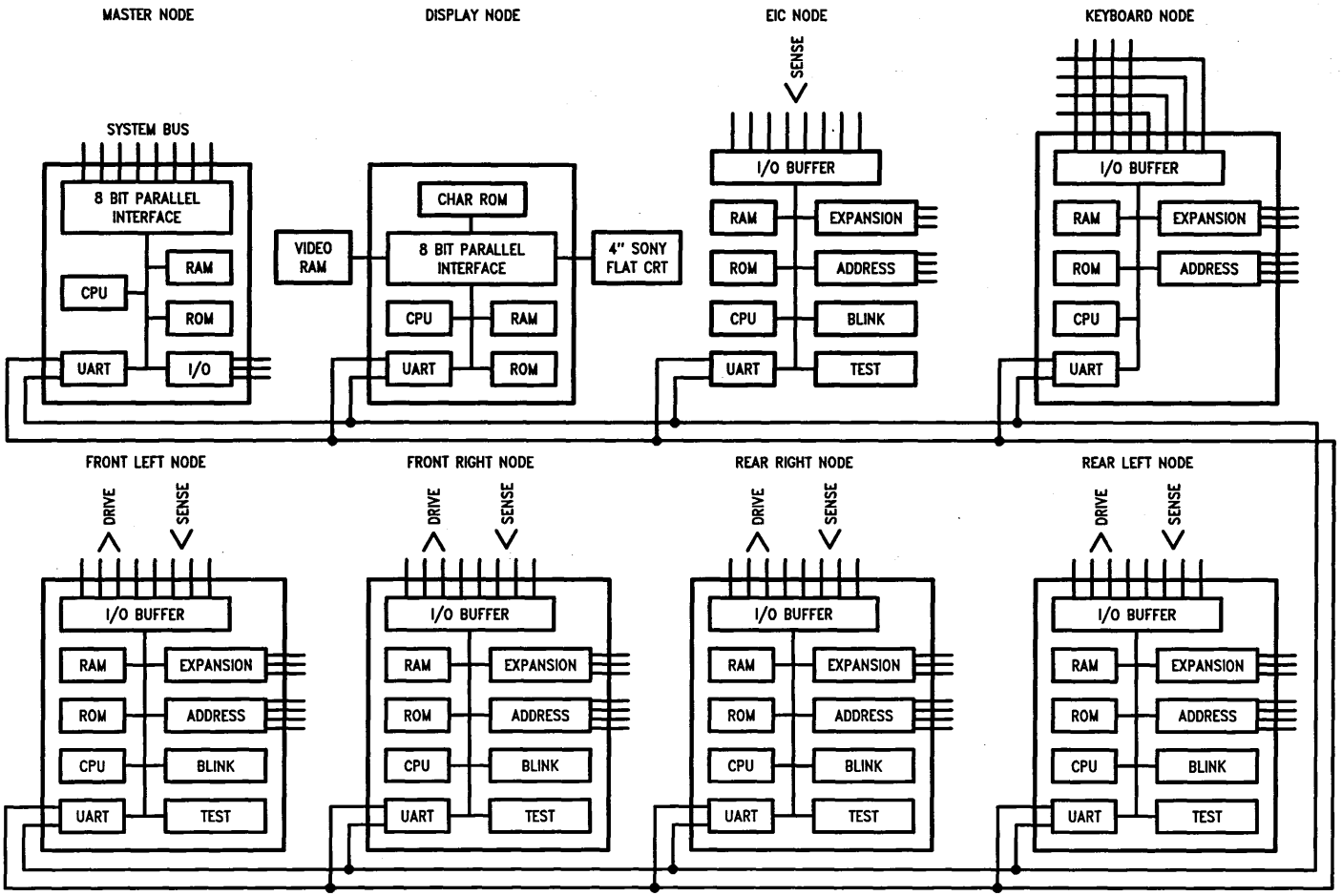
An Electronic Instrument Cluster (EIC) system is a completely independent system. It typically performs all functions associated with the automobile dashboard such as vehicle speed, odometers to accumulate mileage, gauges to display engine temperature, fuel level and so on. It also indicates error conditions such as high engine temperatures, low fuel level etc. The multiplex wiring system uses a standard slave node as a bridge between the two independent systems. The slave node monitors error conditions from the EIC system and passes them to the master node upon request. It becomes relatively simple to allow the master to access all activity in the EIC system via additional commands to the slave node serving the EIC system:

THE COMMUNICATION PROTOCOL

The master unit addresses the remote units sequentially and receives a status reply from each individual node. Data communication is via the standard UART format. It has a start bit, eight data bits, an even parity bit and one stop bit.

Information to be transmitted from the master to a slave node is organized as a frame. Each frame contains the address of destination and command or data. The information in a frame is transmitted as byte format. Address/data differentiation is done by means of a flag. The byte is an address byte if the MSB is set ("1"), otherwise it is a data byte.

Two different types of addressing schemes have been incorporated into the communication protocol; node addressing and class addressing. A class of nodes is formed by grouping together slave nodes with common functions. Commands may be executed either by specific individual nodes or by slave classes. All nodes of the same class execute the command simultaneously. The system implementation at National involved four classes with seven slave nodes per class. So, the total number of nodes possible in this system is 28.



TL/DD/8799–1

FIGURE 1. Block Diagram

The partitioning between the class address and node address reduces the density of bus traffic significantly by eliminating repetative command transmission to individual node class. Lower bus traffic implies that lower transmission bit rate can be used, allowing additional noise immunity. Another advantage of the class addressing is the provision of synchronization for control signals such as HAZARD, LEFT/RIGHT turns.

Error correction is incorporated into the communication protocol. The UART error flags such as PARITY and FRAMING ERRORS protect the system at the physical layer. At the system level, the nodes simply avoid sending an acknowledgement to the master when an error is detected. The master times out and sends the command again.

THE MASTER NODE

The master controller is the heart of the system. Its responsibility is to generate the controlling commands and synchronize the system. It transmits to the remote units and listens to them to get the vehicle status and acts accordingly. Circuit complexity is reduced by implementing extensive software programming in the master controller. This means that the burden is essentially on the master and must be engineered to very high standards of reliability. The device used in the implementation as the master is the COP1430. It is a cost effective 4-bit single chip microcontroller. It features on chip UART which handles asynchronous communications at speeds up to 9600 Baud.

THE SLAVE NODES

The standard slave nodes are based upon the COP413L. The COP413L is a low cost 4-bit microcontroller which may be customized in production. A system such as multiplex wiring requires power consumption to be absolutely minimal. Another basic requirement is that the system should be cost effective. These two facts directed us to use the COP413L at the standard slave node. The COP413L is a low cost (49¢!) low power microcontroller from NSC drawing less than 7 mA at 4.5V to 5.5V. The device contains an 8-bit bidirectional I/O port and a serial expansion port. The CMOS version of COP413L will also be available.

THE DISPLAY NODE

This node can serve as a condition monitoring unit for the vehicle. A considerable quantity of diagnostic information collected from transducers, switches, sensors and various loads are fed to this unit to be displayed on a CRT display. The node is based on a Terminal Management Processor the NS455. The NS455 is a CRT controller on chip. The messages are updated over the serial I/O line by the master controller. The communication format is:

a) The node receives the address.

b) If address matches the local node address, send the copy command

c) Receive new address and execute.

OUTPUT STAGES

The power FETs used for local switching throughout the system are IRF541(4). These N-channel FETs provide much better drive circuit specification as compared to bipolar output stages. They also feature all of the well established advantages of MOSFET such as voltage control, very fast switching, and very low on state resistance. Another advantage is the lower cost as compared to comparibly rated p-channel devices.

TRANSMISSION MEDIUM

A balanced twisted pair is used for bus medium which provides high noise immunity. The transceiver selected for the bus is DS3695 *(Figure 2)*. This device is a high speed differential TRI-STATE® Bus/line transceiver designed to meet EIA standard for multipoint bus transmission. Bus contention or fault situations that cause excessive power dissipation within the device are handled by a standard thermal shutdown circuit, which forces the driver outputs into the high impedance state.

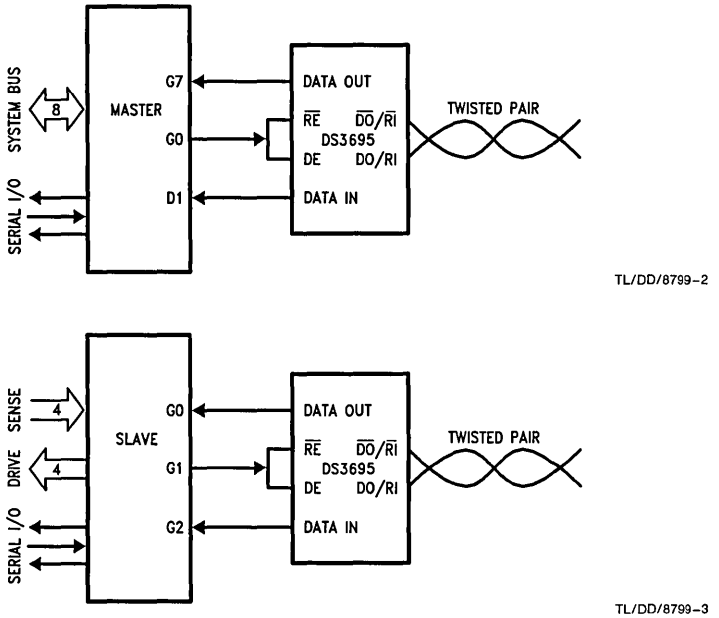


FIGURE 2. Bus Interface

CONCLUSIONS

Multiplex wiring system potentially seems to be a good replacement for conventional wiring system. Reduced complexity, increased flexibility and diagnostic capability could be achieved by incorporating microcontroller devices at nodes within the wiring system. The 4-bit microcontrollers selected are available in a price range, as low as 49¢, that will allow multiplex wiring to compare favorably on a cost-performance basis with the conventional harness.

REFERENCES

1. Michael W. Lowndes and Paul E. V. Phillips, "The Motorcar Multiplex Systems", IEE Conference on Automotive Electronics, 229, England, Nov. 1983.
2. R. F. Robins/W. J. Brittain/M. R. Lunt, "A Car Multiplex Wiring System with Self Coding Control Modules", IEE Conference on Automotive Electronics, 229, Ford Motor Company, UK, Nov. 1983.
3. Booth, J. A., 1983 "Vehicle Interconnection Systems for the Future", IEE Conference on Automotive Electronics, London, Nov. 1983.
4. International Rectifier, HEXFET Databook, 1985.

Dual Tone Multiple Frequency (DTMF)

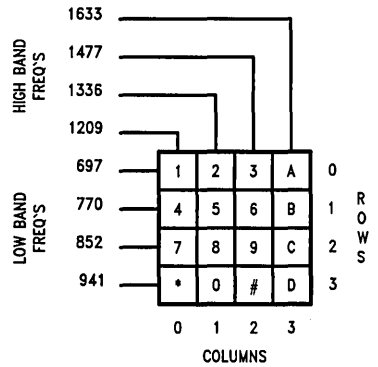
National Semiconductor
Application Note 521
Verne H. Wilson

The DTMF (Dual Tone Multiple Frequency) application is associated with digital telephony, and provides two selected output frequencies (one high band, one low band) for a duration of 100 ms. A benchmark subroutine has been written for the COP820C/840C microcontrollers, and is outlined in detail in this application note. This DTMF subroutine takes 110 bytes of COP820C/840C code, consisting of 78 bytes of program code and 32 bytes of ROM table. The timings in this DTMF subroutine are based on a 20 MHz COP820C/840C clock, giving an instruction cycle time of 1 μs.

The matrix for selecting the high and low band frequencies associated with each key is shown in *Figure 1*. Each key is uniquely referenced by selecting one of the four low band frequencies associated with the matrix rows, coupled with selecting one of the four high band frequencies associated with the matrix columns. The low band frequencies are 697, 770, 852, and 941 Hz, while the high band frequencies are 1209, 1336, 1477, and 1633 Hz. The DTMF subroutine assumes that the key decoding is supplied as a low order hex digit in the accumulator. The COP820C/840C DTMF subroutine will then generate the selected high band and low band frequencies on port G output pins G3 and G2 respectively for a duration of 100 ms.

The COP820C/840C each contain only one timer. The problem is that three different times must be generated to satisfy the DTMF application. These three times are the periods of the two selected frequencies and the 100 ms duration period. Obviously the single timer can be used to generate any one (or possibly two) of the required times, with the program having to generate the other two (or one) times.

The solution to the DTMF problem lies in dividing the 100 ms time duration by the half periods (rounded to the nearest micro second) for each of the eight frequencies, and then examining the respective high band and low band quotients and remainders. The results of these divisions are detailed in Table I. The low band frequency quotients range from 139 to 188, while the high band quotients range from 241 to 326. The observation that only the low band quotients will each fit in a single byte dictates that the high band frequency be produced by the 16 bit (2 byte) COP820C/840C timer running in PWM (Pulse Width Modulation) Mode.



TL/DD/9662–1

FIGURE 1. DTMF Keyboard Matrix

The solution then is to use the program to produce the selected low band frequency as well as keep track of the 100 ms duration. This is achieved by using three programmed register counters R0, R2, and R3, with a backup register R1 to reload the counter R0. These three counters represent the half period, the 100 ms quotient, and the 100 ms remainder associated with each of the four low band frequencies.

The theory of operation in producing the selected low band frequency starts with loading the three counters with values obtained from a RAM table. The half period for the selected frequency is counted out, after which the G2 output bit is toggled. During this half period countout, the quotient counter is decremented. This procedure is repeated until the quotient counter counts out, after which the program branches to the remainder loop. During the remainder loop, the remainder counter counts out to terminate the 100 ms. Following the remainder countout, the G2 and G3 bits are both reset, after which the DTMF subroutine is exited. Great care must be taken in time balancing the half period loop for the selected low band frequency. Furthermore, the toggling of the G2 output bit (achieved with either a set or reset bit instruction) must also be exactly time balanced to maintain the half period time integrity. Local stall loops (consisting of a DRSZ instruction followed by a JP jump back to the DRSZ for a two byte, six instruction cycle loop) are embedded in both the half period and remainder loops. Consequently, the ROM table parameters for the half period and remainder counters are approximately only one sixth of what otherwise might be expected. The program for the half period loop, along with the detailed time balancing of the loop for each of the low band frequencies, is shown in *Figure 2*.

The DTMF subroutine makes use of two 16 byte ROM tables. The first ROM table contains the translation table for the input hex digit into the core vector. The encoding of the hex digit along with the hex digit ROM translation table is shown in Table II. The row and column bits (RR, CC) representing the low band and high band frequencies respectively of the keyboard matrix shown in *Figure 1*, are encoded in

TABLE I. Frequency Half Periods, Quotients, and Remainders

	Freq. Hz	Half Period 0.5P	Half Period in μs	100 ms/0.5P	
				Quotient	Remainder
Low Band Freq.'s	697	717.36	717	139	337
	770	649.35	649	154	54
	852	586.85	587	170	210
	941	531.35	531	188	172
High Band Freq.'s	1209	413.56	414 (256 + 158)	241	226
	1336	374.25	374 (256 + 118)	267	142
	1477	338.52	339 (256 + 83)	294	334
	1633	306.18	306 (256 + 50)	326	244

the two upper and two lower bits of the hex digit respectively. Consequently, the format for the hex digit bits is RRCC, so that the input byte in the accumulator will consist of 0000RRCC. The program changes this value into 1101RRCC before using it in setting up the address for the hex digit ROM translation table.

The core vectors from the hex digit ROM translation table consist of a format of TT00XX00, where the two T (Timer) bits select one of four high band frequencies, while the two X bits select one of four low band frequencies. The core vector is transformed into four different inputs for the second ROM table. This transformation of the core vector is shown in Table III. The core vector transformation produces a timer vector 1100TT00 (T), and three programmed counter vectors for R1, R2, and R3. The formats for the three counter vectors are 1100XX11 (F), 1100XX10 (Q), and 1100XX01 (R) for R1, R2, and R3 respectively. These four vectors produced from the core vector are then used as inputs to the second ROM table. One of these four vectors (the T vector) is a function of the T bits from the core vector, while the other three vectors (F, Q, R) are a function of the X bits. This correlates to only one parameter being needed for the timer (representing the selected high band frequency), while three parameters are needed for the three counters (half period, 100 ms quotient, 100 ms remainder) associated with the low band frequency and 100 ms duration. The frequency parameter ROM translation table, accessed by the T, F, Q, and R vectors, is shown in Table IV.

Program			Bytes/Cycle	Conditional Cycles		Cycles	Total Cycles
	LD	B,#PORTGD	2/3				
	LD	X,#R1	2/3				
LUP1:	LD	A,[X−]	1/3			3	
	IFBIT	2,[B]	1/1			1	
	JP	BYP1	1/3	3	1		
	X	A,[X+]	1/3		3		
	SBIT	2,[B]	1/1		1		
	JP	BYP2	1/3		3		
BYP1:	NOP		1/1	1			
	RBIT	2,[B]	1/1	1			
	X	A,[X+]	1/3	3			
BYP2:	DRSZ	R2	1/3 DECREMENT			3	
	JP	LUP2	1/3 Q COUNT			3	
	JP	FINI	1/3				
LUP2:	DRSZ	R0	1/3 DECREMENT		3	3	
	JP	LUP2	1/3 F COUNT		3	1	
	NOP		1/1			1	
	LD	A,[X]	1/3			3	
	IFEQ	A,#104	2/2			2	
	JP	LUP1	1/3		1	3	31
	NOP		1/1		1		
	IFEQ	A,#93	2/2		2		
BACK:	JP	LUP1	1/3	1	3		35
	JP	BACK	1/3	3			
				3			39

Table IV Frequency	Stall Loop	Total Cycles	Half Period
((114 − 1)	x 6)	+ 39	= 717
((104 − 1)	x 6)	+ 31	= 649
((93 − 1)	x 6)	+ 35	= 587
((83 − 1)	x 6)	+ 39	= 531

FIGURE 2. Time Balancing for Half Period Loop

TABLE II. Hex Digit ROM Translation Table

	0	1	2	3
ROW	697 Hz	770 Hz	852 Hz	941 Hz
COLUMN	1209 Hz	1336 Hz	1477 Hz	1633 Hz

ADDRESS *	DATA (HEX)	KEYBOARD *
0xD0	000	1
0xD1	004	2
0xD2	008	3
0xD3	00C	A
0xD4	040	4
0xD5	044	5
0xD6	048	6
0xD7	04C	B
0xD8	080	7
0xD9	084	8
0xDA	088	9
0xDB	08C	C
0xDC	0C0	*
0xDD	0C4	0
0xDE	0C8	#
0xDF	0CC	D

* HEX DIGIT IS RRCC,
WHERE R = ROW #
AND C = COLUMN #
- - - EXAMPLE: KEY 3 IS ROW #0,
COLUMN #2, SO HEX DIGIT
IS 0010 = 2
RRCC

TABLE III. Core Vector Translation

CORE VECTOR - TT00XX00 - - - - - - - - - -

```
                                *
                               * *
                              * * *
```

TIMER VECTOR	TIMER	T	1100TT00
HALF PERIOD VECTOR	R1	F	1100XX11
QUOTIENT VECTOR	R2	Q	1100XX10
REMAINDER VECTOR	R3	R	1100XX01

TABLE IV. Frequency Parameter ROM Translation Table

T - TIMER F - FREQUENCY Q - QUOTIENT R - REMAINDER

ADDRESS	DATA (DEC)	VECTOR
0xC0	158	T
0xC1	53	R
0xC2	140	Q
0xC3	114	F
0xC4	118	T
0xC5	6	R
0xC6	155	Q
0xC7	104	F
0xC8	83	T
0xC9	32	R
0xCA	171	Q
0xCB	93	F
0xCC	50	T
0xCD	25	R
0xCE	189	Q
0xCF	83	F

In summary, the input hex digit selects one of 16 core vectors from the first ROM table. This core vector is then transformed into four other vectors (T, F, Q, R), which in turn are used to select four parameters from the second ROM table. These four parameters are used to load the timer, and the respective half period, quotient, and remainder counters. The first ROM table (representing the hex digit matrix table) is arbitrarily placed starting at ROM location 01D0, and has a reference setup with the ADD A,#0D0 instruction. The second ROM table (representing the frequency parameter table) must be placed starting at ROM location 01C0 (or 0xC0) in order to minimize program size, and has reference setups with the OR A,#0C3 instruction for the F vector and with the OR A,#0C0 instruction for the T vector.

The three parameters associated with the two X bits of the core vector require a multi-level table lookup capability with the LAID instruction. This is achieved with the following section of code in the DTMF subroutine:

```
        LD      B,#R1
        LD      X,#R4
        X       A,[X]
LUP:    LD      A,[X]
        LAID
        X       A,[B+]
        DRSZ    R4
        IFBNE   #4
        JP      LUP
```

This program code loads the F frequency vector into R4, and then decrements the vector each time around the loop. This successive loop decrementation of the R4 vector changes the F vector into the Q vector, and then changes the Q vector into the R vector. This R4 vector is used to access the ROM table with the LAID instruction. The X pointer references the R4 vector, while the B pointer is incremented each time around the loop after it has been used to store away the three selected ROM table parameters (one per loop). These three parameters are stored in sequential RAM locations R1, R2, and R3. The IFBNE test instruction is used to skip out of the loop once the three selected ROM table parameters have been accessed and stored away.

The timer is initialized to a count of 15 so that the first timer underflow and toggling of the G3 output bit (with timer PWM mode and G3 toggle output selected) will occur at the same time as the first toggling of the G2 output bit. The half period counts for the high band frequencies range from 306 to 414, so these values minus 256 are stored in the timer section of the second ROM table. The selected value from this frequency ROM table is then stored in the lower half of the timer autoreload register, while a 1 is stored in the upper half. The timer is selected for PWM output mode and started with the instruction LD [B],#0B0 where the B pointer is selecting the CNTRL register at memory location 0EE.

The DTMF subroutine for the COP820C/840C uses 110 bytes of code, consisting of 78 bytes of program code and 32 bytes of ROM table. A program routine to sequentially call the DTMF subroutine for each of the 16 hex digit inputs is supplied with the listing for the DTMF subroutine.

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     1
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF

  1                ;DTMF PROGRAM FOR COP820C/840C          VERNE H. WILSON
  2                ;                                            8/25/87
  3                ;DTMF - DUAL TONE MULTIPLE FREQUENCY
  4                ;
  5                ;PROGRAM NAME: DTMF.MAC
  6                ;
  7                          .TITLE DTMF
  8                ;
  9                ;******* THE DTMF SUBROUTINE CONTAINS 110 BYTES *******
 10                ; ***** THE DTMF SUBROUTINE TIMES OUT IN 100MSEC *****
 11                ;    ** FROM THE FIRST TOGGLE OF THE G2/G3 OUTPUTS **
 12                ;       *** BASED ON A 20 MHZ COP820C/840C CLOCK ***
 13                ;
 14                ;G PORT IS USED FOR THE TWO OUTPUTS
 15                ;     -    HIGH BAND (HB) FREQUENCY OUTPUT ON G3
 16                ;     -    LOW BAND (LB) FREQUENCY OUTPUT ON G2
 17                ;
 18                ;TIMER COUNTS OUT
 19                ;     -    HB FREQUENCIES
 20                ;
 21                ;PROGRAM COUNTS OUT
 22                ;     -    LB FREQUENCIES
 23                ;     -    100 MSEC DIVIDED BY LB HALF PERIOD QUOTIENT
 24                ;     -    100 MSEC DIVIDED BY LB HALF PERIOD REMAINDER
 25                ;
 26                ;FORMAT FOR THE 16 HEX DIGIT MATRIX VECTOR IS 1101RRCC,
 27                ;    WHERE  -  RR IS ROW SELECT (LB FREQUENCIES)
 28                ;           -  CC IS COLUMN SELECT (HB FREQUENCIES)
 29                ;
 30                ;FORMAT FOR THE 16 CORE VECTORS FROM THE MATRIX SELECT
 31                ;    TABLE IS TT00XX00, WHERE  -  TT IS HB SELECT
 32                ;                                 XX IS LB SELECT
 33                ;
 34                ;FREQUENCY VECTORS (HB & LB) FOR FREQ PARAMETER TABLE
 35                ;    MADE FROM CORE VECTORS
 36                ;
 37                ;HB FREQUENCY VECTORS(4) END WITH 00 FOR TIMER COUNTS,
 38                ;    WHERE VECTOR FORMAT IS 1100TT00
 39                ;
 40                ;LB FREQUENCY VECTORS(12) END WITH:
 41                ;    11 FOR HALF PERIOD LOOP COUNTS,
 42                ;         WHERE VECTOR FORMAT IS 1100XX11
 43                ;    10 FOR 100 MSEC DIVIDED BY HALF PERIOD QUOTIENTS,
 44                ;         WHERE VECTOR FORMAT IS 1100XX10
 45                ;    01 FOR 100 MSEC DIVIDED BY HALF PERIOD REMAINDERS,
 46                ;         WHERE VECTOR FORMAT IS 1100XX01
 47                ;
 48                ;HEX DIGIT MATRIX TABLE AT HEX 01D* (OPTIONAL LOCATION,
 49                ;    DEPENDING ON  'ADD  A,#0D0' INST. IMMEDIATE VALUE)
 50                ;
 51                ;FREQ PARAMETER TABLE AT HEX 01C*  (REQUIRED LOCATION)
```

TL/DD/9662-2

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     2
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF


                          .FORM
                ;
                ;MAGIC:        CORE VECTOR
                ;                TT00XX00
                ;
                ;     TIMER      T     TT00
                ;     R1         F     XX11
                ;     R2         Q     XX10
                ;     R3         R     XX01
                ;
                ;DECLARATIONS:
        00D0       PORTLD = 0D0          ; PORTL DATA REG
        00D1       PORTLC = 0D1          ; PORTL CONFIG REG
        00D4       PORTGD = 0D4          ; PORTG DATA REG
        00D5       PORTGC = 0D5          ; PORTG CONFIG REG
        00DC        PORTD = 0DC          ; PORTD REG
        00EA      TIMERLO = 0EA          ; TIMER LOW COUNTER
        00EE        CNTRL = 0EE          ; CONTROL REG
        00EF          PSW = 0EF          ; PROC STATUS WORD
        00F0           R0 = 0F0          ; LB FREQ LOOP COUNTER
        00F1           R1 = 0F1          ; LB FREQ LOOP COUNT
        00F2           R2 = 0F2          ; LB FREQ Q COUNT
        00F3           R3 = 0F3          ; LB FREQ R COUNT
        00F4           R4 = 0F4          ; LB FREQ TABLE VECTOR
                ;
0000 DD2F       START:      LD       SP,#02F        ; HEX DIGIT MATRIX
0002 BCD1FF                 LD       PORTLC,#0FF    ; 1  2  3  A
0005 BCD080                 LD       PORTLD,#080    ; 4  5  6  B
0008 DEDC                   LD       B,#PORTD       ; 7  8  9  C
000A 9E00                   LD       [B],#0         ; *  0  #  D
000C AE         LOOP:       LD       A,[B]          ; DTMF TEST LOOP
000D 3160                   JSR      DTMF           ; HEX MATRIX DIGIT
000F DEDC                   LD       B,#PORTD       ; TO SUBROUTINE IS
0011 AE                     LD       A,[B]          ; OUTPUT TO PORTD
0012 9405                   ADD      A,#5           ; DO WILL TOGGLE
0014 A6                     X        A,[B]          ; FOR EACH CALL OF
0015 6C                     RBIT     4,[B]          ; DTMF SUBROUTINE
0016 9DD0                   LD       A,PORTLD       ; PORTL OUTPUTS
0018 A1                     SC                      ; PROVIDE SYNC
0019 B0                     RRC      A              ; OUTPUT ORDER IS
001A 9CD0                   X        A,PORTLD       ; 1,5,9,D,4,8,#,A,
001C EF                     JP       LOOP           ; 7,0,3,B,*,2,6,C
                ;
                ;
```

TL/DD/9662-3

```
 97      0160                .=0160
 98                  ;
 99 0160 DED5        DTMF:   LD          B,#PORTGC
100 0162 9B3F               LD          [B-],#03F
101 0164 6B                 RBIT        3,[B]           ; OPTIONAL
102 0165 6A                 RBIT        2,[B]           ; OPTIONAL
103                 ;
104 0166 94D0               ADD         A,#0D0
105 0168 A4                 LAID                        ; DIGIT MATRIX TABLE
106                 ;
107 0169 5F                 LD          B,#0
108 016A A6                 X           A,[B]
109 016B AE                 LD          A,[B]
110 016C 97C3               OR          A,#0C3
111 016E DEF1               LD          B,#R1
112 0170 DCF4               LD          X,#R4
113 0172 B6                 X           A,[X]
114 0173 BE         LUP:    LD          A,[X]
115 0174 A4                 LAID                        ; LB FREQ TABLES
116 0175 A2                 X           A,[B+]          ;   (3 PARAMETERS)
117 0176 C4                 DRSZ        R4
118 0177 44                 IFBNE       #4
119 0178 FA                 JP          LUP
120                 ;
121 0179 5F                 LD          B,#0
122 017A AE                 LD          A,[B]
123 017B 65                 SWAP        A
124 017C 97C0               OR          A,#0C0
125 017E A4                 LAID                        ; HB FREQ TABLE
126 017F DEEA               LD          B,#TIMERLO      ;   (1 PARAMETER)
127 0181 9A0F               LD          [B+],#15
128 0183 9A00               LD          [B+],#0
129 0185 A2                 X           A,[B+]
130 0186 9A01               LD          [B+],#1
131 0188 9EB0               LD          [B],#0B0        ; START TIMER PWM
132                 ;
133 018A DED4               LD          B,#PORTGD
134 018C DCF1               LD          X,#R1
135                 ;
136 018E BB         LUP1:   LD          A,[X-]
137 018F 72                 IFBIT       2,[B]           ; TEST LB OUTPUT
138 0190 03                 JP          BYP1
139 0191 B2                 X           A,[X+]
140 0192 7A                 SBIT        2,[B]           ; SET LB OUTPUT
141 0193 03                 JP          BYP2
142 0194 B8         BYP1:   NOP
143 0195 6A                 RBIT        2,[B]           ; RESET LB OUTPUT
144 0196 B2                 X           A,[X+]
145 0197 C2         BYP2:   DRSZ        R2              ; DECR. QUOT. COUNT
146 0198 01                 JP          LUP2
147 0199 0C                 JP          FINI            ; Q COUNT FINISHED
148                 ;
149 019A C0         LUP2:   DRSZ        R0              ; DECR. F COUNT
150 019B FE                 JP          LUP2            ; LB (HALF PERIOD)
151                 ;
152 019C B8                 NOP                         ; **********
153 019D BE                 LD          A,[X]           ; BALANCE
154 019E 9268               IFEQ        A,#104          ; LB FREQUENCY
155 01A0 ED                 JP          LUP1            ; HALF PERIOD
156                 ;                                   ; RESIDUE
157 01A1 B8                 NOP                         ; DELAY FOR
158 01A2 925D               IFEQ        A,#93           ; EACH OF 4
159 01A4 E9         BACK:   JP          LUP1            ; LB FREQ'S
160 01A5 FE                 JP          BACK            ; **********
161                 ;
162 01A6 C3         FINI:   DRSZ        R3              ; DECR. REM. COUNT
163 01A7 FE                 JP          FINI            ; R CNT NOT FINISHED
164                 ;
165 01A8 BDEE6C             RBIT        4,CNTRL         ; STOP TIMER
166 01AB 6B                 RBIT        3,[B]           ; CLR HB OUTPUT
167 01AC 6A                 RBIT        2,[B]           ; CLR LB OUTPUT
168                 ;
169 01AD 8E                 RET
170                 ;
```

TL/DD/9662-4

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     4
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF


                                    .FORM
                ;
                ;FREQUENCY AND 100MSEC PARAMETER TABLE
        01C0                .=01C0
                ;
 01C0 9E                    .BYTE     158             ; T
 01C1 35                    .BYTE      53             ; R
 01C2 8C                    .BYTE     140             ; Q
 01C3 72                    .BYTE     114             ; F
 01C4 76                    .BYTE     118             ; T
 01C5 06                    .BYTE       6             ; R
 01C6 9B                    .BYTE     155             ; Q
 01C7 68                    .BYTE     104             ; F
 01C8 53                    .BYTE      83             ; T
 01C9 20                    .BYTE      32             ; R
 01CA AB                    .BYTE     171             ; Q
 01CB 5D                    .BYTE      93             ; F
 01CC 32                    .BYTE      50             ; T
 01CD 19                    .BYTE      25             ; R
 01CE BD                    .BYTE     189             ; Q
 01CF 53                    .BYTE      83             ; F
                ;
                ;DIGIT MATRIX TABLE
        01D0                .=01D0
                ;                                                  ROW   COL
 01D0 00                    .BYTE     000             ; 1         1     1
 01D1 04                    .BYTE     004             ; 2         1     2
 01D2 08                    .BYTE     008             ; 3         1     3
 01D3 0C                    .BYTE     00C             ; A         1     4
 01D4 40                    .BYTE     040             ; 4         2     1
 01D5 44                    .BYTE     044             ; 5         2     2
 01D6 48                    .BYTE     048             ; 6         2     3
 01D7 4C                    .BYTE     04C             ; B         2     4
 01D8 80                    .BYTE     080             ; 7         3     1
 01D9 84                    .BYTE     084             ; 8         3     2
 01DA 88                    .BYTE     088             ; 9         3     3
 01DB 8C                    .BYTE     08C             ; C         3     4
 01DC C0                    .BYTE     0C0             ; *         4     1
 01DD C4                    .BYTE     0C4             ; 0         4     2
 01DE C8                    .BYTE     0C8             ; #         4     3
 01DF CC                    .BYTE     0CC             ; D         4     4
                ;
                                    .END
```

TL/DD/9662-5

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF

SYMBOL TABLE

 B       00FE          BACK    01A4          BYP1    0194          BYP2    0197
 CNTRL   00EE          DTMF    0160          FINI    01A6          LOOP    000C
 LUP     0173          LUP1    018E          LUP2    019A          PORTD   00DC
 PORTGC  00D5          PORTGD  00D4          PORTLC  00D1          PORTLD  00D0
 PSW     00EF  *       R0      00F0          R1      00F1          R2      00F2
 R3      00F3          R4      00F4          SP      00FD          START   0000  *
 TIMERL  00EA          X       00FC


MACRO TABLE


   NO WARNING LINES

   NO ERROR LINES

  139  ROM BYTES USED

SOURCE CHECKSUM = 99A7
OBJECT CHECKSUM = 03E1

INPUT   FILE C:DTMF.MAC
LISTING FILE C:DTMF.PRN
OBJECT  FILE C:DTMF.LM
```

TL/DD/9662-6

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

Section 4

HPC Family

Section 4 Contents

National Semiconductor

The 16-Bit HPC™ Family: Optimized for Performance

Key Features

- World's first 16-bit CMOS microcontroller
- World's fastest CMOS microcontroller
- 134 ns instruction-cycle time at 30 MHz
- Full 16-bit architecture and implementation
- 64 kbyte address space
- High code efficiency with single-byte, multiple-function instructions
- 16 x 16-bit multiply, 32 x 16-bit divide
- Eight vectored interrupt sources
- Watchdog logic monitors
- 16-bit timer/counters
- Up to 52 general-purpose high-speed I/O lines
- On-chip ROM to 8 kbytes
- On-chip RAM to 256 bytes
- On-chip peripherals
  - — DMA
  - — HDLC
  - — Timers
  - — Input-capture registers
  - — A/D converter
  - — UART
  - — User-programmable memory
  - — High speed SRAM
  - — Gate array
- M²CMOS fabrication
- MICROWIRE/PLUS™ serial interface
- ROMless versions available
- Wide operating voltage range: +3V to +5.5V
- Military temp range available (−55°C to +125°C)
- MIL-STD-883C versions available
- 68-pin PGA, PLCC, and LDCC packages

National's High Performance Controller (HPC) family is not only the world's first 16-bit CMOS microcontroller family, but also the world's fastest.

Currently operating at a clock rate of 30 MHz, the HPC's 2-micron geometry is fabricated in scalable M²CMOS™, allowing die-shrinks to 1.25 microns and, ultimately, to submicron levels. Meaning the HPC will be operating at much higher frequencies in the future.

The HPC is designed for high-performance applications. With its 134 ns instruction cycle and its 16 x 16-bit multiply and 32 x 16-bit divide, the HPC is appropriate for compute-intensive environments that used to be the sole domain of the microprocessor.

The HPC is ideal, for example, for signal conditioning applications. The HPC's high throughput helps eliminate external components from typical signal processing/control circuits, and allows key parts of the application to be implemented in software rather than hardware.

This not only reduces system cost and development time, but also increases the flexibility and market life of the product.

At the same time, because the HPC has a control-oriented architecture, important functions are still implemented in hardware, providing critical performance advantages unavailable in a pure-software solution, such as a general microprocessor-based design.

It is this powerful performance capability that, when combined with the wide range of peripheral functions that are available (such as UARTs, A/D converters, and HDLC protocol controllers), make the HPC a true systems solution on a chip.

The Powerful HPC Core

The HPC is an "application-specific" microcontroller.

Based on a common, high-performance CPU "core", each HPC family member can be "customized" to meet the exact needs of a particular application.

The core, based on a microprocessor-like von Neumann architecture, contains seven key functional elements:

1. Arithmetic Logic Unit (ALU)
2. 6 working registers
3. 8 interrupts
4. 3 timers
5. Control logic
6. Watchdog circuitry
7. MICROWIRE/PLUS interface

The internal data paths, registers, timers, and ALU are all 16 bits wide.

So the HPC can directly address up to 64 kbytes of "external" memory.

The external data bus, however, is dynamically configurable as 8 or 16 bits, allowing it to efficiently interface with a variety of peripheral devices.

Flexible Peripheral Support

The HPC core can support a full range of peripheral functions:

- High-level Data Link Control (HDLC) for ISO-standard data communications

Flexible Peripheral Support (Continued)

- Universal Asynchronous Receiver/Transmitters (UARTs) for full-duplex, 300/1200/2400/9600-baud serial communications
- High-Speed Outputs and Pulse-Width Modulated (PWM) timers for efficient external interfaces
- User-programmable memory
- Analog-to-Digital (A/D) converters for interfacing "real-world" inputs

Plus:

- Up to 64 kbytes of direct-addressable memory
- Up to 52 I/O ports on a 68-pin package

Efficient Instruction Set

The HPC family achieves much of its performance through its unique, highly optimized instruction set. Unlike the instruction set of a typical microprocessor, the HPC instruction set is designed for maximum code efficiency. Because ROM-space is necessarily limited on a single-chip solution, programs must be compact and economical.

The HPC instruction set supports nine addressing modes, like a high-performance 16-bit microprocessor. And each instruction in the set is designed to execute a number of individual functions, so the same operations can be executed with tighter code.

As a result, the typical HPC instruction cycle is only 134 ns at 30 MHz. And the typical HPC 16-bit multiply or divide takes less than 4 μs.

To achieve the same level of performance in other 16-bit and high-end 8-bit microcontrollers, as indicated by recent benchmark studies, would require up to *two times the memory space* as the HPC.

Low Power Operation

The HPC uses power as efficiently as it uses memory space.

The HPC draws only 20 mA of current at 17 MHz. And its even less at lower clock rates.

The HPC can also operate effectively at input voltages as low as +3.0V, which further reduces power consumption.

In addition, the HPC has two software-selectable power-down modes:

1. IDLE, which stops all operations except for the oscillator and one timer, thereby maintaining all RAM, registers, and I/O in a static state, cuts current drain to 2 mA.

2. HALT, which stops all operations including the oscillator and timers, but holds RAM, registers, and I/O stable, cuts current drain to only 20 μA.

Key Applications

- Signal conditioning/processing/control
- Automotive systems
- Data processing
- Telecommunications
- Military
- Embedded controllers
- Medical
- Factory automation
- Industrial control
- Compute-intensive environments
- High-end control
- Tape and disk drives
- Security systems
- Laser printers
- SCSI control

High Level Language Support

A C compiler is already available for software development on standard platforms: the IBM PC running DOS or UNIX® or the DEC™ VAX™ running VMS™ or UNIX.

With powerful tools such as these, the HPC can be quickly and efficiently programmed for any high-performance application.

HPC Family of Microcontrollers

Commercial Temp Version 0°C to +70°C	Industrial Temp Version −40°C to +85°C	Military Temp Version −55°C to +125°C	Memory		Features						
			ROM (Bytes)	RAM (Bytes)	I/O		Interrupt	Stack	Timer Base Counters	Size (Pins)	Other*
					I/O Pins	Serial I/O					
HPC46003	HPC36003	HPC16003	ROMless	256	52	YES	8 Sources	In RAM	8	68	4 ICR's
HPC46004	HPC36004	HPC16004	ROMless	512	52	YES	8 Sources	In RAM	8	68	4 ICR's
HPC46064	HPC36064	HPC16064	16.0k	512	52	YES	8 Sources	In RAM	8		4 ICR's
HPC46083	HPC36083	HPC16083	8.0k	256	52	YES	8 Sources	In RAM	8	68	4 ICR's
HPC46104	HPC36104	HPC16104	ROMless	512	52	YES	8 Sources	In RAM	8		4 ICR's & 8 CH A/D
HPC46164	HPC36164	HPC16164	16.0k	512	52	YES	8 Sources	In RAM	8	68	4 ICR's & 8 CH A/D
HPC46400	HPC36400	HPC16400	N/A	256	52	YES	8 Sources	In RAM	4	68	HDLC & DMA
HPC46900	HPC36900	HPC16900	N/A							68	PEARL

*ICR = Input Capture Registers

HDLC = High-Level Data Link Control

PEARL = Port Expanded and Recreation Logic

National Semiconductor

PRELIMINARY

HPC16083/HPC26083/HPC36083/HPC46083/ HPC16003/HPC26003/HPC36003/HPC46003 High-Performance microControllers

General Description

The HPC16083 and HPC16003 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC16083 has 8k bytes of on-chip ROM. The HPC16003 has no on-chip ROM and is intended for use with external memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

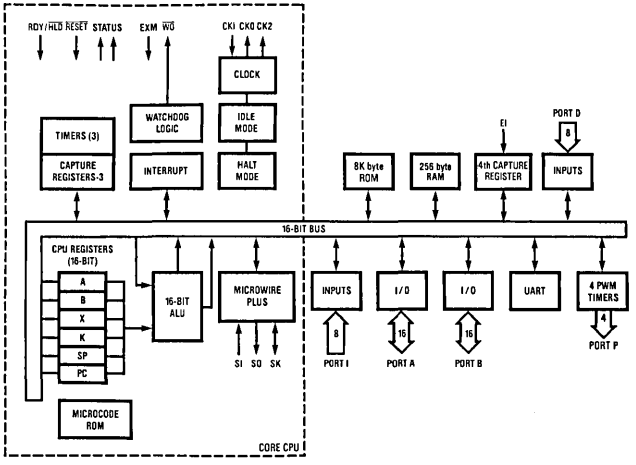
The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC16083" is used throughout this datasheet to refer to the HPC16083, HPC16043 and HPC16003 devices unless otherwise specified.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LCC, LDCC, PGA and TapePak® packages.

Features

- HPC family—core features:
  - — 16-bit architecture, both byte and word
  - — 16-bit data bus, ALU, and registers
  - — 64k bytes of external memory addressing
  - — FAST—240 ns for fastest instruction when using 17.0 MHz clock, 134 ns at 30 MHz
  - — High code efficiency—most instructions are single byte
  - — 16 x 16 multiply and 32 x 16 divide
  - — Eight vectored interrupt sources
  - — Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
  - — MICROWIRE/PLUS serial I/O interface
  - — CMOS—very low power with two power save modes: IDLE and HALT
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 8k bytes of ROM, 256 bytes of RAM on chip
- ROMless version available (HPC16003)
- Commercial (0°C to +70°C), industrial (−40°C to +85°C), automotive (−40°C to +105°C) and military (−55°C to +125°C) temperature ranges

Block Diagram (HPC16083 with 8k ROM shown)



TL/DD/8801-1

17 MHz
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46043/HPC46003, −40°C to +85°C for HPC36083/HPC36043/HPC36003, −40°C to +105°C for HPC26083/HPC26043/HPC26003, −55°C to +125°C for HPC16083/HPC16043/HPC16003

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 17.0 MHz (Note 1)		30	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		3.5	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 17.0 MHz, (Note 1)		3.0	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		0.35	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		200	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		75	μA
INPUT VOLTAGE LEVELS $\overline{RESET}$, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA, V_{CC} = 5.0V	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	I_{OH} = −6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{RESET}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}, CKI driven to V_{IH} and V_{IL}, with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

17 MHz

AC Electrical Characteristics V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	17.0	MHz
$t_{C1} = 1/f_C$	Clock Period	59		ns
$t_C = 2/f_C$	Timing Cycle	118		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	50		ns
t_{DC1C2R}	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F}	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		50	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		50	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	23		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	24		ns
$t_{WAIT} = t_C = WS$	Wait State Period	118		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		892	kHz
t_{XIN}	Pulse Width for Timer Inputs	177		ns
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.12	MHz

Read Cycle Timing with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{RD}$ Falling Edge	24		ns
$t_{RW} = \frac{1}{2} t_C + WS - 10$	$\overline{RD}$ Pulse Width	167		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{RD}$	0	75	ns
$t_{ACC} = t_C + WS - 55$	Address Valid to Input Data Valid		181	ns
$t_{RD} = \frac{1}{2} t_C + WS - 65$	$\overline{RD}$ Falling Edge to Input Data Valid		112	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	111		ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 pF, other Outputs C_L = 80 pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurements will be skewed.

Note: Minimum and Maximum values are calculated from maximum operating frequency.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not yet tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a high confidence level.

17 MHz

Write Cycle Timing with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{WR}$ Falling Edge	54		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{WR}$ Pulse Width	192		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{WR}$	24		ns
$t_V = \frac{1}{2} t_C + WS - 15$	Data Valid before Rising Edge of $\overline{WR}$	162		ns

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		98	ns
$t_{RWP} = t_C$	RDY Pulse Width	118		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	129		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	128		ns
$t_{HAD} = \frac{7}{4} t_C + 50$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		257	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		218*	ns
t_{BF}	Bus Float before Falling Edge on $\overline{HLDA}$	0		ns
$t_{BE} = \frac{3}{4} t_C + 50$	Bus Enable from Rising Edge of $\overline{HLDA}$		139	ns

***Note:** t_{HAE} may be as long as $(3t_C + 4ws + 72t_C + 90)$ depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value tested is for the optimal case.

UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{URD}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{URD}$	10		ns
t_{RPW}	$\overline{URD}$ Pulse Width	100		ns
t_{OE}	$\overline{URD}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{URD}$ to Output Data Valid	5	35	ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Rising Edge of $\overline{URD}$		70	ns
t_{WDW}	$\overline{UWR}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{UWR}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{UWR}$	15		ns
t_A	$\overline{WRRDY}$ Delay from Rising Edge of $\overline{UWR}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

30 MHZ
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 30.0 MHz (Note 1)		60	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		3.5	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 30.0 MHz, (Note 1)		6	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		0.35	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		200	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		75	μA
INPUT VOLTAGE LEVELS $\overline{RESET}$, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA, V_{CC} = 5.0V	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH_4}	ST1 and ST2 Drive	I_{OH} = −6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{RESET}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}, CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

30 MHZ

AC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	30	MHz
$t_{C1} = 1/f_C$	Clock Period	33		ns
$t_C = 2/f_C$	Timing Cycle	67		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	24		ns
t_{DC1C2R}	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F}	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} tC + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		37	ns
$t_{DC2ALEF} = \frac{1}{4} tC + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		37	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	10		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	12		ns
$t_{WAIT} = t_C = WS$	Wait State Period	67		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.58	kHz
t_{XIN}	Pulse Width for Timer Inputs	100		ns
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.58	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		3.75	MHz

Read Cycle Timing with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{RD}$ Falling Edge	12		ns
$t_{RW} = \frac{1}{2} t_C + WS - 10$	$\overline{RD}$ Pulse Width	90		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{RD}$	0	35	ns
$t_{ACC} = t_C + WS - 33$	Address Valid to Input Data Valid		100	ns
$t_{RD} = \frac{1}{2} t_C + WS - 25$	$\overline{RD}$ Falling Edge to Input Data Valid		75	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	62		ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 pF, other Outputs C_L = 80 pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurements will be skewed.

Note: Minimum and Maximum values are calculated from maximum operating frequency.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not yet tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a high confidence level.

Note: These AC specifications are subject to change based on final device characterization. Please contact the factory for updated information.

30 MHz

Write Cycle Timing with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{WR}$ Falling Edge	28		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{WR}$ Pulse Width	102		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{WR}$	12		ns
$t_V = \frac{1}{2} t_C + WS - 15$	Data Valid before Rising Edge of $\overline{WR}$	85		ns

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		33	ns
$t_{RWP} = t_C$	RDY Pulse Width	67		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	90		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	77		ns
$t_{HAD} = \frac{7}{4} t_C + 50$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		167	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$	167	167*	ns
t_{BF}	Bus Float before Falling Edge on $\overline{HLDA}$	0		ns
$t_{BE} = \frac{3}{4} t_C + 50$	Bus Enable from Rising Edge of $\overline{HLDA}$		100	ns

***Note:** t_{HAE} may be as long as $(3t_C + 4ws + 72t_C + 90)$ depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value is for the optimal case.

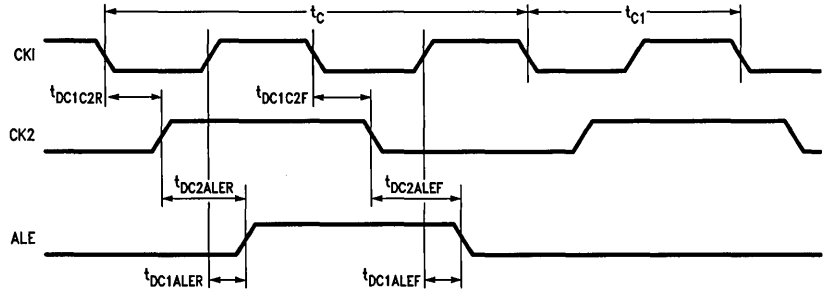
UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{URD}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{URD}$	10		ns
t_{RPW}	$\overline{URD}$ Pulse Width	100		ns
t_{OE}	$\overline{URD}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{URD}$ to Output Data Valid	5	35	ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Rising Edge of $\overline{URD}$		70	ns
t_{WDW}	$\overline{UWR}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{UWR}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{UWR}$	15		ns
t_A	$\overline{WRRDY}$ Delay from Rising Edge of $\overline{UWR}$		70	ns

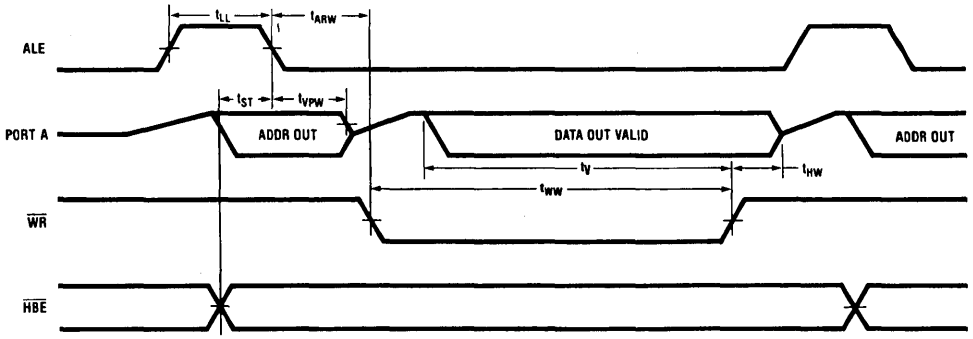
Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

Timing Waveforms

CKI, CK2, ALE Timing Diagram

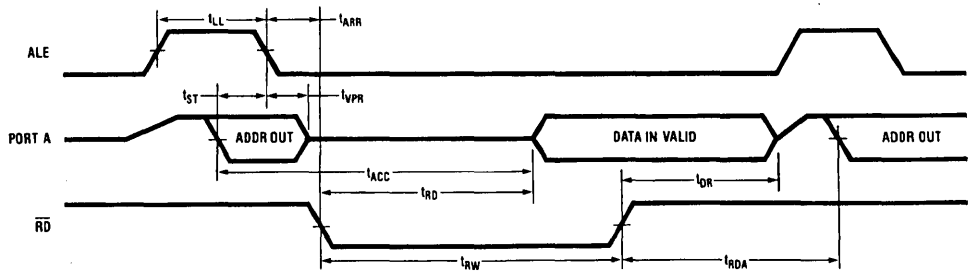


TL/DD/8801-33



TL/DD/8801-3

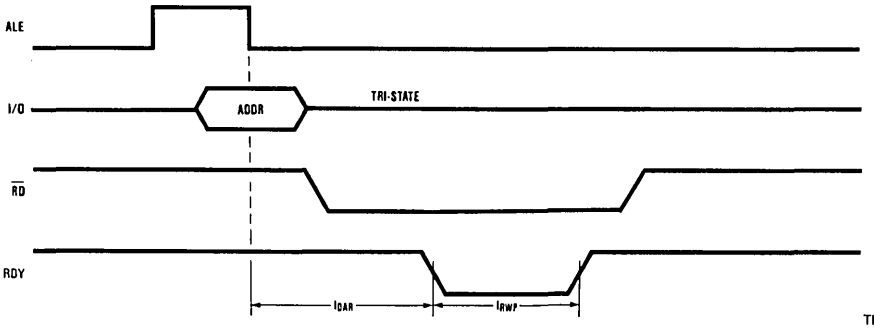
FIGURE 1. Write Cycle



TL/DD/8801-4

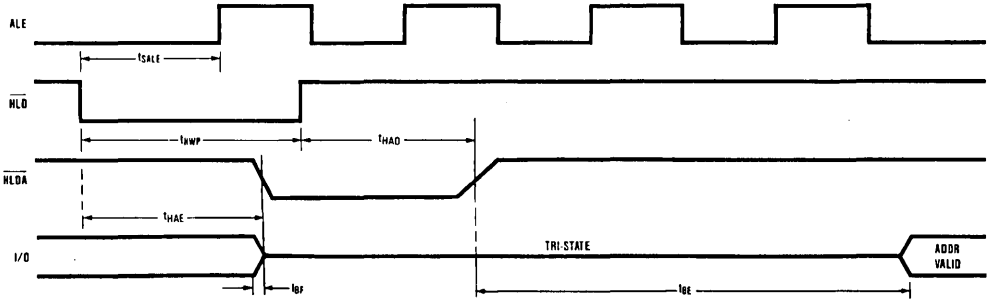
FIGURE 2. Read Cycle

Timing Waveforms (Continued)



TL/DD/8801-5

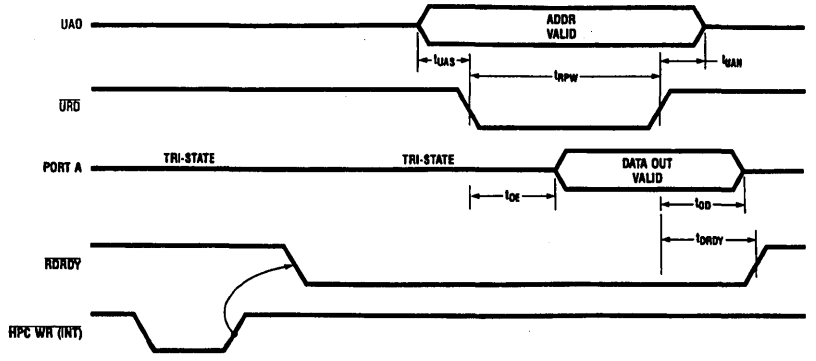
FIGURE 3. Ready Mode Timing



TL/DD/8801-6

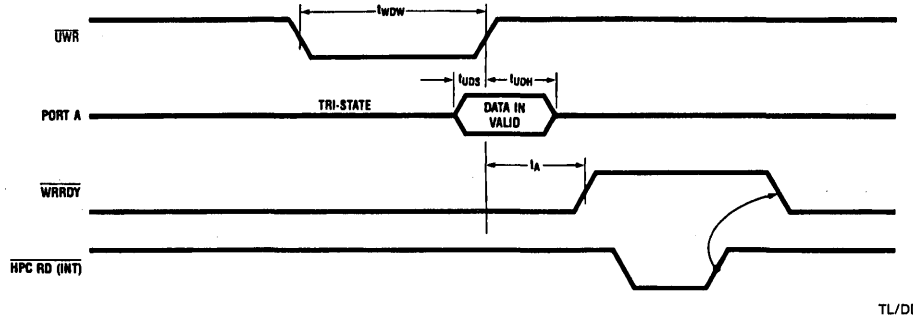
FIGURE 4. Hold Mode Timing

Timing Waveforms (Continued)



TL/DD/8801-9

FIGURE 5. UPI Read Timing



TL/DD/8801-10

FIGURE 6. UPI Write Timing

Pin Descriptions

The HPC16083 is available in 68-pin PLCC, LCC, LDCC, PGA and TapePak packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{HLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	$\overline{\text{WRRDY}}$	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RDRDY}}$	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	$\overline{\text{WR}}$	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output/Input (sampled at reset)
B15:	$\overline{\text{RD}}$	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/$\overline{\text{URD}}$
I3:	INT3	Maskable Interrupt/Input Capture/$\overline{\text{UWR}}$
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4

Pin Descriptions (Continued)

through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V_{CC1} and V_{CC2} — Positive Power Supply

GND — Ground for On-Chip Logic

DGND — Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI — The Chip System Clock Input

CKO — The Chip System Clock Output (inversion of CKI)

Pins CKI and CKO are usually connected across an external crystal.

CK2 — Clock Output (CKI divided by 2)

OTHER PINS

$\overline{WO}$ — This is an active low open drain output that signals an illegal situation has been detected by the Watch Dog logic.

ST1 — Bus Cycle Status Output: indicates first opcode fetch.

ST2 — Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).

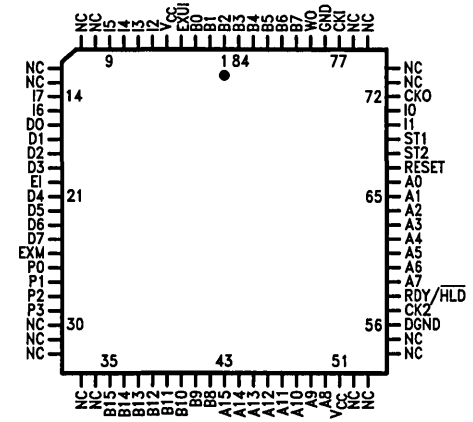
$\overline{RESET}$ — is an active low input that forces the chip to restart and sets the ports in a TRI-STATE® mode.

RDY/$\overline{HLD}$ has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.

NC — (no connection) do not connect anything to this pin.

EXM — External memory enable (active high) disables internal ROM and maps it to external memory.

TapePak Package



TL/DD/8801-34

Top View

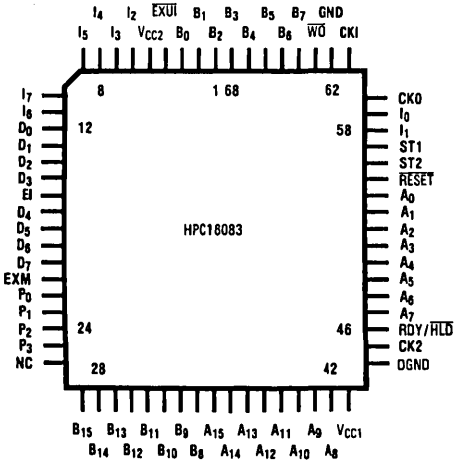
Order Number HPC16083T
Available in TapePak

EI — External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.

$\overline{EXUI}$ — External interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 (Active Low).

Connection Diagrams

Plastic, Leadless and Leaded Chip Carriers

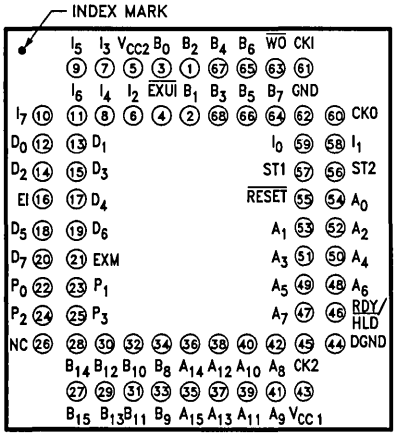


TL/DD/8801-11

Top View

Order Number HPC16083E or V
See NS Package Number E68B or V68A

Pin Grid Array Pinout



TL/DD/8801-12

Top View
(looking down on component side of PC Board)

Order Number HPC16083EL or HPC16083U
See NS Package Number EL68A or U68A

4

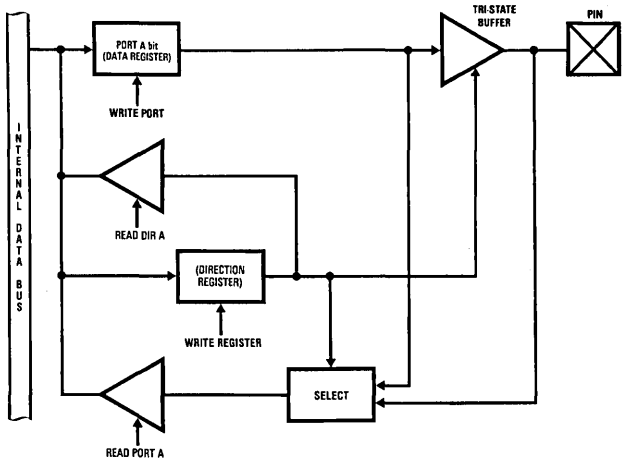
Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 7*), consists of a data register and a direction register. Port B (see *Figures 8, 9, 10*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

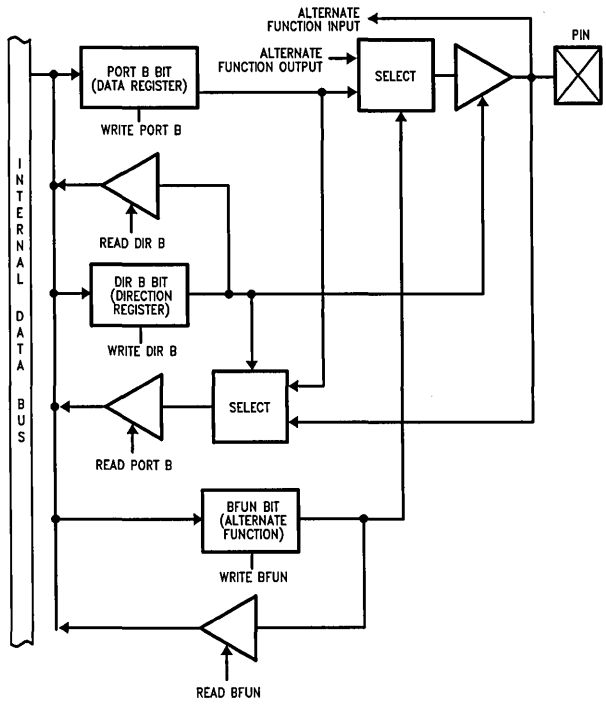
A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.



TL/DD/8801–13

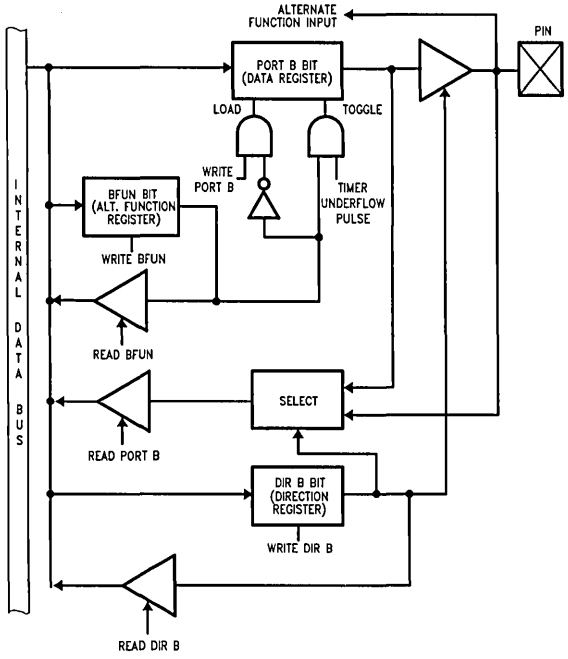
FIGURE 7. Port A: I/O Structure



TL/DD/8801–14

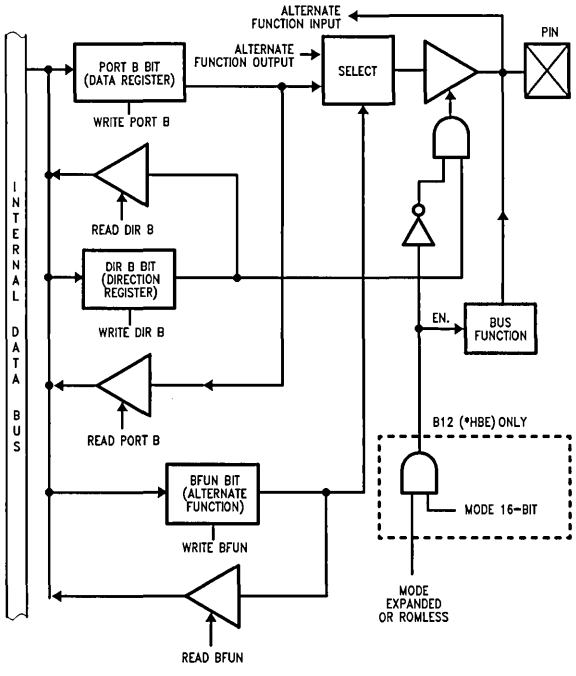
FIGURE 8. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

Ports A & B (Continued)



TL/DD/8801-15

FIGURE 9. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)



TL/DD/8801-16

FIGURE 10. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC16083 has four operating modes. The ROMless HPC16003 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC16083 is E000 to FFFF (8k bytes). The HPC16003 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC16003 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range and on-chip RAM and Register range, and the "illegal address detection" feature of the Watchdog logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the Watchdog logic is disabled. The EA bit should be set to "1" by software when using the HPC16003 to disable the "illegal address detection" feature of Watchdog.

All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, $\overline{RD}$, $\overline{WR}$ and $\overline{HBE}$. The High Byte Enable pin ($\overline{HBE}$) is used in 16-bit mode to select high order memory bytes. The $\overline{RD}$ and $\overline{WR}$ signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling $\overline{HBE}$ high at reset. If $\overline{HBE}$ is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC16083 and HPC16003.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC16083 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC16083 functions as a self-contained microcomputer (see *Figure 11*) with all memory (RAM and ROM) on-chip. It can address internal memory only, consisting of 8k bytes of ROM (E000 to FFFF) and 512 bytes of on-chip RAM and registers (0000 to 02FF). The "illegal address detection" feature of the Watchdog is enabled in the Single-Chip Normal mode and a Watchdog Output ($\overline{WO}$) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC16083 to address external memory in addition to the on-chip ROM and RAM (see Table II). Watchdog illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC16083 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 8k bytes of external memory may be used with the HPC16083 (see Table II). The Watchdog circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

EXPANDED ROMLESS MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of Watchdog is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

TABLE II. HPC16083 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	E000:FFFF on-chip
Expanded Normal	0	1	E000:FFFF on-chip 0200:DFFF off-chip
Single-Chip ROMless	1	0	E000:FFFF off-chip
Expanded ROMless	1	1	0200:FFFF off-chip

Note: In all operating modes, the on-chip RAM and Registers (0000:01FF) may be accessed.

HPC16003 Operating Modes

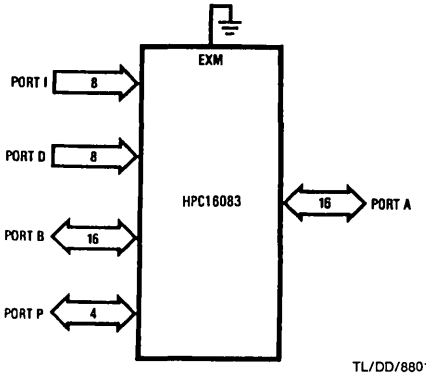
EXPANDED ROMLESS MODE (HPC16003)

Because the HPC16003 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC16003 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between F000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the Watchdog logic.

TABLE III. HPC16003 Operating Modes

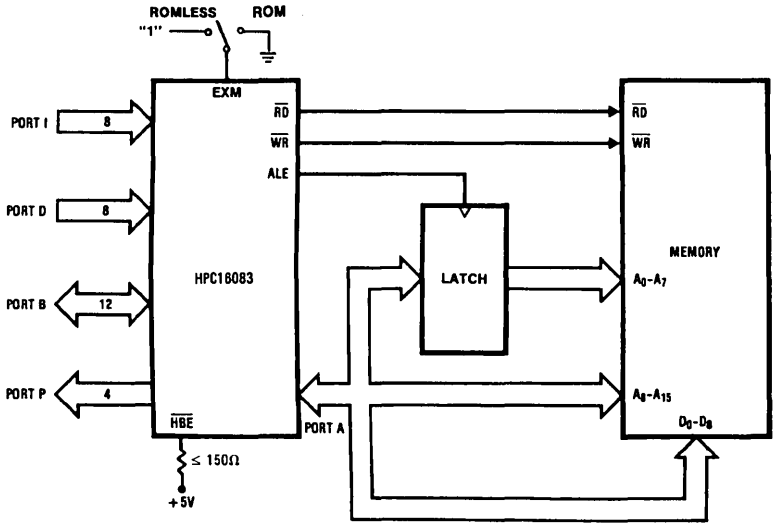
Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0200:FFFF off-chip

Note: The on-chip RAM and Registers (0000:01FF) of the HPC16003 may be accessed at all times.



TL/DD/8801-17

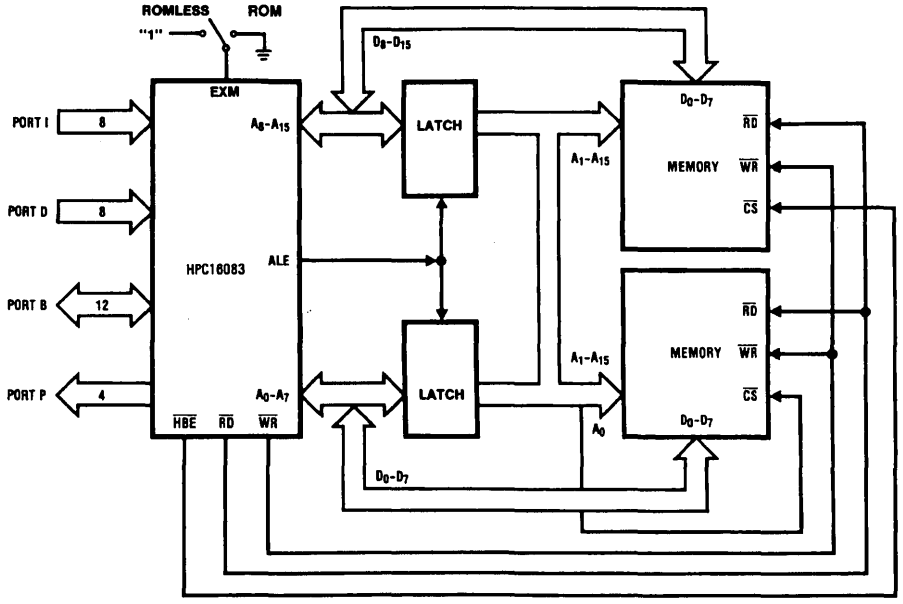
FIGURE 11. Single-Chip Mode



TL/DD/8801-18

FIGURE 12. 8-Bit External Memory

Operating Modes (Continued)



TL/DD/8801-19

FIGURE 13. 16-Bit External Memory

Wait States

The HPC16083 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC16083: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16083 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16083 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the $\overline{\text{RESET}}$ or the NMI. The $\overline{\text{RESET}}$ input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16083 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the $\overline{\text{RESET}}$ or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC16083 to resume normal operation.

HPC16083 Interrupts

Complex interrupt handling is easily accomplished by the HPC16083's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table IV.

TABLE IV. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
$FFFF:FFFE	$\overline{\text{RESET}}$	0
$FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
$FFFB:FFFA	External interrupt on I2 pin	2
$FFF9:FFF8	External interrupt on I3 pin	3
$FFF7:FFF6	External interrupt on I4 pin	4
$FFF5:FFF4	Overflow on internal timers	5
$FFF3:FFF2	Internal on the UART transmit/receive complete or external on EXUI	6
$FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC16083 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table IV. The interrupt on $\overline{\text{RESET}}$ has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the $\overline{\text{RESET}}$, which is serviced immediately. $\overline{\text{RESET}}$ and $\overline{\text{EXUI}}$ are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt ($\overline{\text{EXUI}}$) is shared with the UART interrupt. This interrupt is level-low sensitive. To select this interrupt disable the ERI and ETI UART interrupt bits in the ENUI register. To select the UART interrupt leave this pin floating or tie it high.

Interrupt Control Registers

The HPC16083 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

$\overline{\text{RESET}}$ and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to be serviced, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16083 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

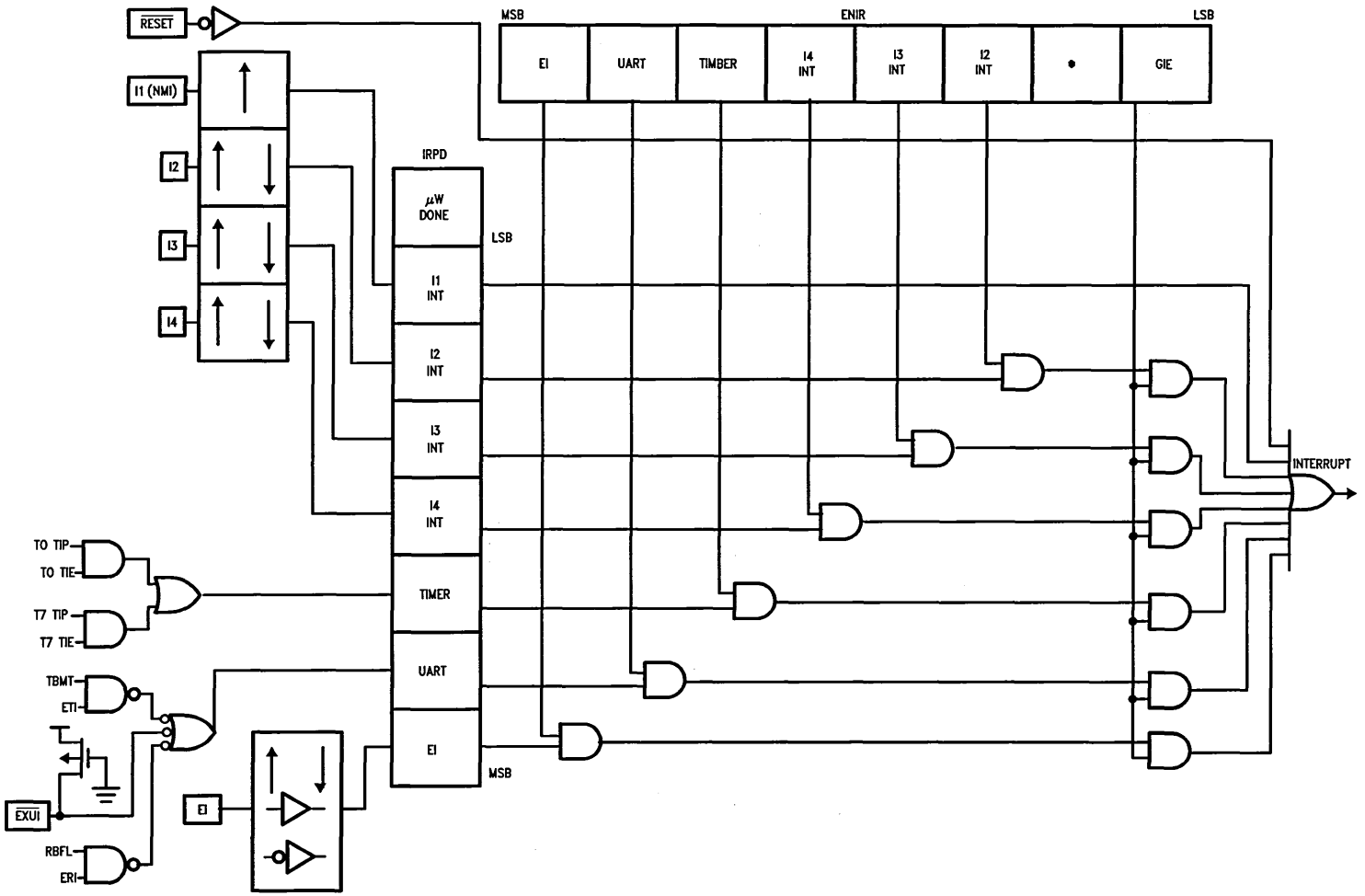
Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 14* shows the Interrupt Enable Logic.

$\overline{\text{RESET}}$

The $\overline{\text{RESET}}$ input initializes the processor and sets ports A and B in the TRI-STATE condition and port P in the LOW state. $\overline{\text{RESET}}$ is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between F000 and FFFF when using the HPC16003.



TL/DD/8801–20

FIGURE 14. Block Diagram of Interrupt Logic

Timer Overview

The HPC16083 contains a powerful set of flexible timers enabling the HPC16083 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16083 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watchdog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 15*).

The HPC16083 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 16*).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

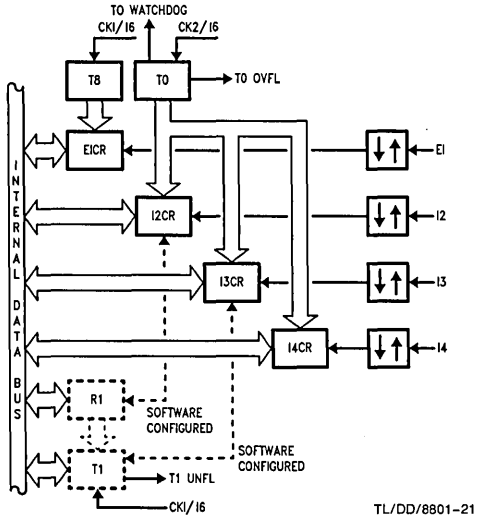


FIGURE 15. Timers T0, T1 and T8 with Four Input Capture Registers

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16083 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 16*).

Timer/register pairs 4–7 form four identical units which can generate synchronous outputs on port P (see *Figure 17*).

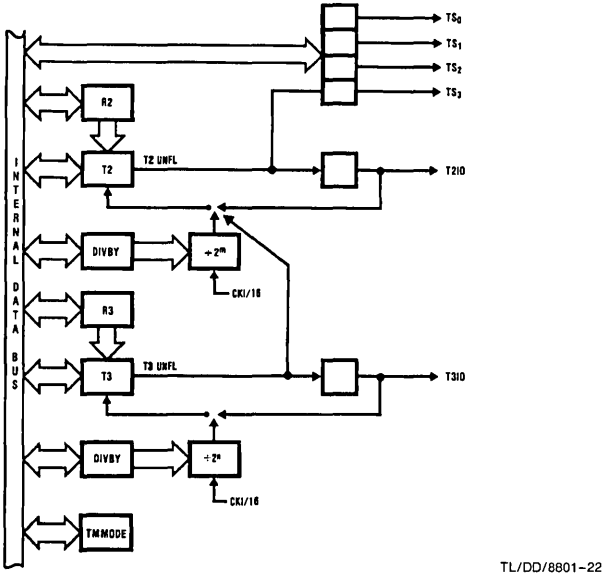
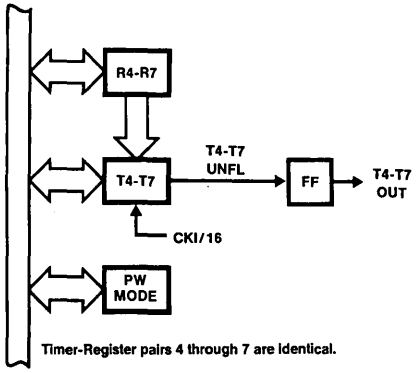


FIGURE 16. Timers T2–T3 Block

Timer Overview (Continued)



TL/DD/8801-23

FIGURE 17. Timers T4–T7 Block

Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to ½ the frequency of the source used for clocking the timer.

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16083.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

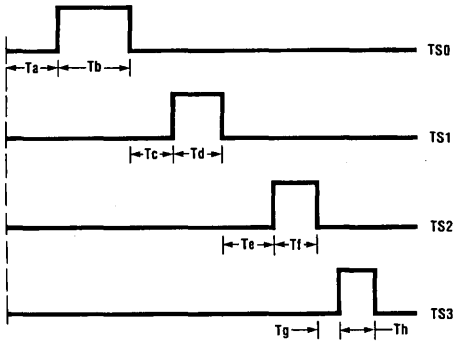
TL/DD/8801-24

FIGURE 18. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 19* is an example of synchronous pulse train generation.

Watchdog Logic

The Watchdog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watchdog logic are potentially infinite loops and illegal addresses. Should the Watchdog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the Watchdog Output ($\overline{WO}$) pin low. The $\overline{WO}$ pin is an open drain output and can be connected to the $\overline{RESET}$ or NMI inputs or to the users external logic.



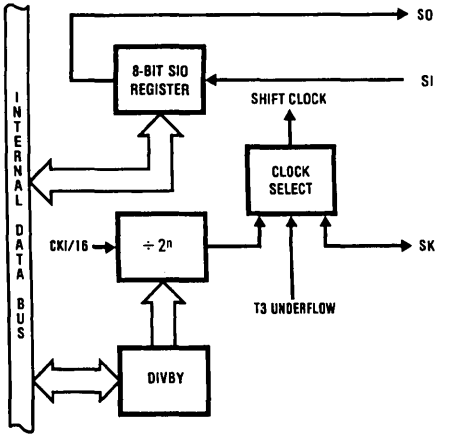
TL/DD/8801-25

FIGURE 19. Synchronous Pulse Generation

***Note:** See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 20*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.



TL/DD/8801-26

FIGURE 20. MICROWIRE/PLUS

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS Operation

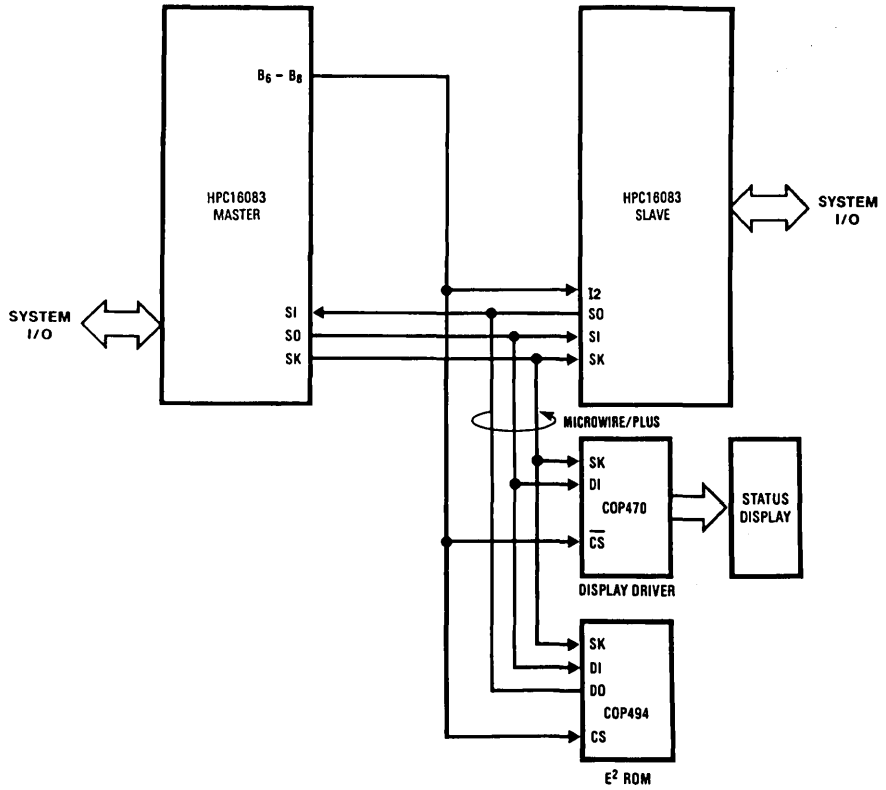
The HPC16083 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16083 is the master or slave. The shift clock is generated when the HPC16083 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16083 is configured as a slave. When the HPC16083 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 21 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC16083 microcontrollers interconnected to other MICROWIRE peripherals. HPC16083 #1 is set up as the master and initiates all data transfers. HPC16083 #2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of a VF display controlled by the COP470 display driver. The data to be displayed is sent serially to the COP470 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC16083 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.



TL/DD/8801–27

FIGURE 21. MICROWIRE/PLUS Application

HPC16083 UART

The HPC16083 contains a software programmable UART. The UART (see *Figure 22*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

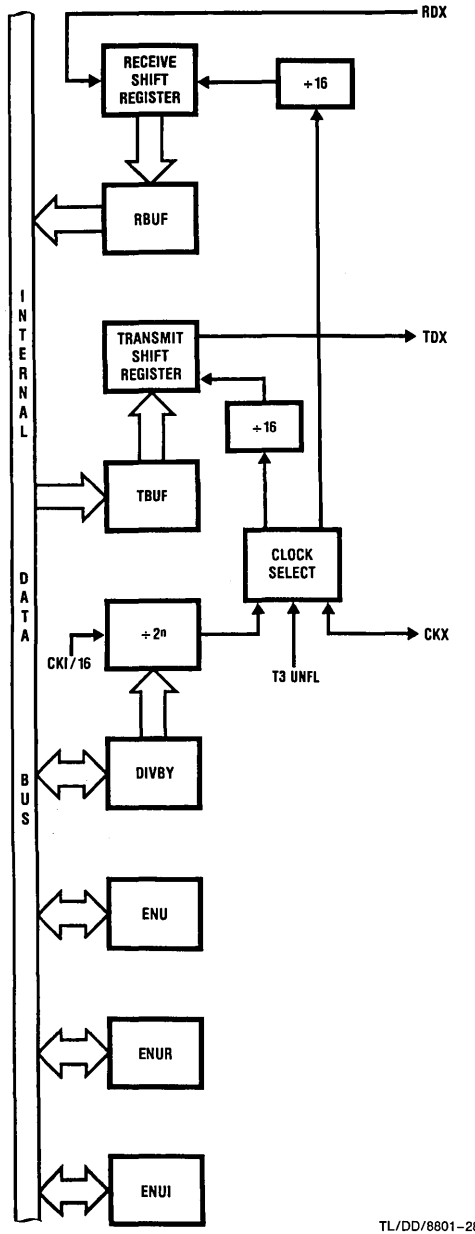
The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC16083 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16083 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16083 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16083 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.



TL/DD/8801–28

FIGURE 22. UART Block Diagram

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC16083 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC16083's and set up systems with very high data exchange rates. Another area of application could be where a HPC16083 is programmed as an intelligent peripheral to a host system such as the Series 32000® microprocessor. *FIGURE 23* illustrates how a HPC16083 could be used an an intelligent peripherial for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe ($\overline{URD}$), a Write Strobe ($\overline{UWR}$), a Read Ready Line ($\overline{RDRDY}$), a Write Ready Line ($\overline{WRRDY}$) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The $\overline{URD}$ and $\overline{UWR}$ inputs may be used to interrupt the HPC16083. The $\overline{RDRDY}$ and $\overline{WRRDY}$ outputs may be used to interrupt the host processor.

The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC16083 is the data bus. UPI can only be used if the HPC16083 is in the Single-Chip mode.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16083 supports shared memory access with two pins. The pins are the RDY/$\overline{HLD}$ input pin and the $\overline{HLDA}$ output pin. The user can software select either the Hold or Ready function by the state of a control bit. The $\overline{HLDA}$ output is multiplexed onto port B.

The host uses DMA to interface with the HPC16083. The host initiates a data transfer by activating the $\overline{HLD}$ input of the HPC16083. In response, the HPC16083 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{HLDA}$) from the HPC16083 indicating that the sytem bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16083 resumes normal operations.

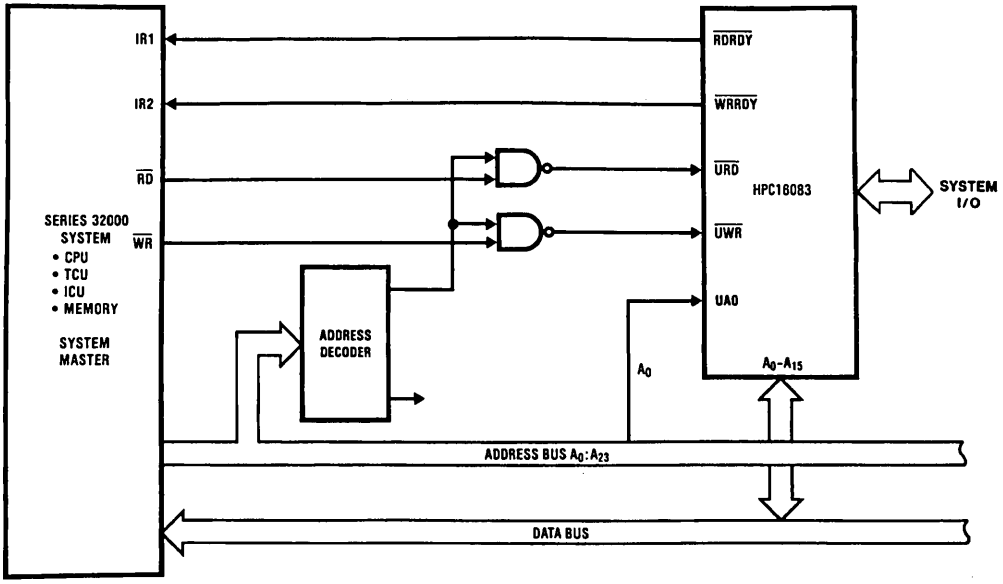
FIGURE 24 illustrates an application of the shared memory interface between the HPC16083 and a Series 32000 system.

Memory

The HPC16083 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 8 kbytes of ROM and 256 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16083 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

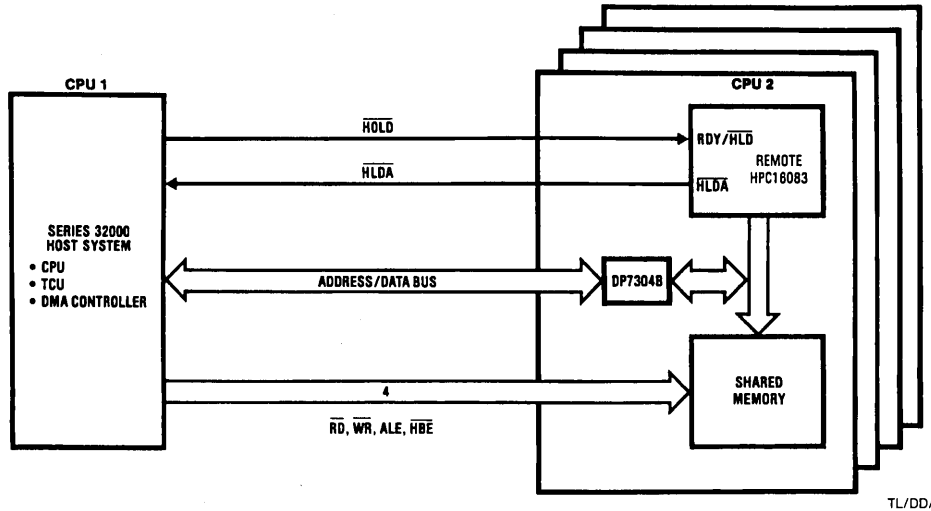
The HPC16083 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table V.



TL/DD/8801-29

FIGURE 23. HPC16083 as a Peripheral: (UPI Interface to Series 32000 Application)

Shared Memory Support (Continued)



TL/DD/8801–30

FIGURE 24. Shared Memory Application: HPC16083 Interface to Series 32000 System

TABLE V. HPC16083 Memory Map

FFFF:FFF0	Interrupt Vectors	USER MEMORY
FFEF:FFD0	JSRP Vectors	
FFCF:FFCE		
: :	On-Chip ROM	
E001:E000		
DFFF:DFFE		
: :	External Expansion	
0201:0200	Memory	
01FF:01FE		USER RAM
: :	On-Chip RAM	
01C1:01C0		
0195:0194	Watchdog Address	Watchdog Logic
0192	T0CON Register	Timer Block T0:T3
0191:0190	TMMODE Register	
018F:018E	DIVBY Register	
018D:018C	T3 Timer	
018B:018A	R3 Register	
0189:0188	T2 Timer	
0187:0186	R2 Register	
0185:0184	I2CR Register/ R1	
0183:0182	I3CR Register/ T1	
0181:0180	I4CR Register	
015E:015F	EICR	Timer Block T4:T7
015C	EICON	
0153:0152	Port P Register	
0151:0150	PWMODE Register	
014F:014E	R7 Register	
014D:014C	T7 Timer	
014B:014A	R6 Register	
0149:0148	T6 Timer	
0147:0146	R5 Register	
0145:0144	T5 Timer	
0143:0142	R4 Register	
0141:0140	T4 Timer	

0128	ENUR Register	UART
0126	TBUF Register	
0124	RBUF Register	
0122	ENUI Register	
0120	ENU Register	
0104	Port D Input Register	
00F5:00F4	BFUN Register	PORTS A & B CONTROL
00F3:00F2	DIR B Register	
00F1:00F0	DIR A Register / IBUF	
00E6	UPIC Register	UPI CONTROL
00E3:00E2	Port B	PORTS A & B
00E1:00E0	Port A / OBUF	
00DE	Microcode ROM Dump	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00DD:00DC	HALT Enable Register	
00D8	Port I Input Register	
00D6	SIO Register	
00D4	IRCD Register	
00D2	IRPD Register	
00D0	ENIR Register	
00CF:00CE	X Register	HPC CORE REGISTERS
00CD:00CC	B Register	
00CB:00CA	K Register	
00C9:00C8	A Register	
00C7:00C6	PC Register	
00C5:00C4	SP Register	
00C3:00C2	(reserved)	
00C0	PSW Register	
00BF:00BE		USER RAM
: :	On-Chip RAM	
0001:0000		

HPC16083 CPU

The HPC16083 CPU has a 16-bit ALU and six 16-bit registers

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16083 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	MA + Meml → MA carry → C
ADC	Add with carry	MA + Meml + C → MA carry → C
ADDS	Add short imm8	MA + imm8 → MA carry → C
DADC	Decimal add with carry	MA + Meml + C → MA (Decimal) carry → C
SUBC	Subtract with carry	MA − Meml + C → MA carry → C
DSUBC	Decimal subtract w/carry	MA − Meml + C → MA (Decimal) carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C
DIVD	Divide Double Word (unsigned)	(X & MA)/Meml → MA, rem → X, 0 → K, carry → C
IFEQ	If equal	Compare MA & Meml, Do next if equal
IFGT	If greater than	Compare MA & Meml, Do next if MA > Meml
AND	Logical and	MA and Meml → MA
OR	Logical or	MA or Meml → MA
XOR	Logical exclusive-or	MA xor Meml → MA
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	Mem + 1 → Mem
DECSZ	Decrement, skip if 0	Mem − 1 → Mem, Skip next if Mem = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Meml → MA
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ±1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP
POP	Pop Stack to Memory	SP−2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A,B±1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B,imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A − 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP),SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP),SP+2 → SP,PC+ # → PC (#is +1025 to −1023)
JSRL	Jump subroutine long	PC → W(SP),SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +257 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP−2 → SP,W(SP) → PC
RETSK	Return then skip next	SP−2 → SP,W(SP) → PC, & skip
RETI	Return from interrupt	SP−2 → SP,W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8 or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
Meml is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data
imm8 is 8-bit immediate data only

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir. (B)	Reg Indir. (X)	Direct	Indir.	Index	Immed.	Direct *	Direct **	Immed. *	Immed. **
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16083 has been designed to be extremely code-efficient. The HPC16083 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16083, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16083 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16083 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16083 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16083 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information system package contains:
DIAL-A-HELPER Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customers problem by actually getting customers system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

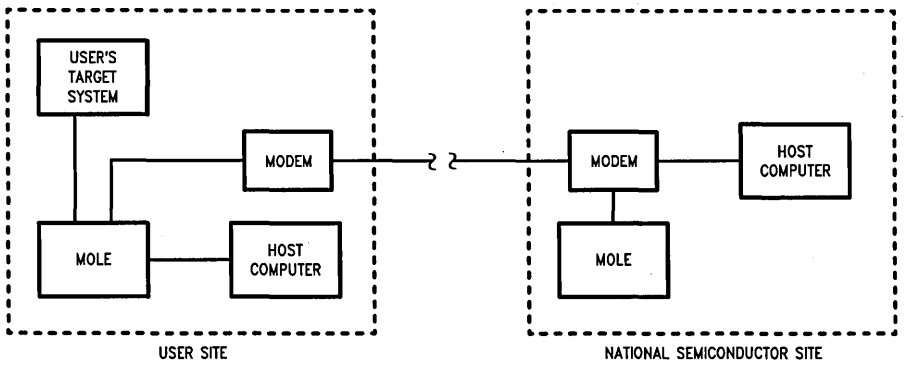
Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBMR	Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	4244105883001
	424410897-001	Users Manual		424410897-001

Development Support (Continued)

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 Baud
Set-Up: Length: 8-bit
Parity: None
Stop Bit: 1
Operation: 24 hrs, 7 days

DIAL-A-HELPER

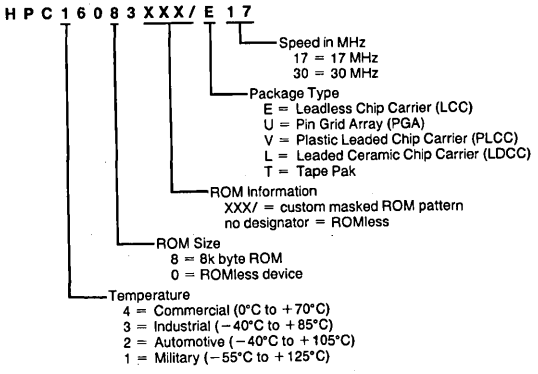


TL/DD/8801-32

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16083 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/8801-31

FIGURE 8. HPC Family Part Numbering Scheme

Examples

HPC46003E17 — ROMless, Commercial temp. (0°C to 70°C), LCC
HPC16083XXX/U17 — 8k masked ROM, Military temp. (−55°C to +125°C), PGA
HPC26083XXX/V17 — 8k masked ROM, Automotive temp. (−40°C to +105°C), PLCC

National Semiconductor

ADVANCE INFORMATION

HPC16164/HPC26164/HPC36164/HPC46164 HPC16104/HPC26104/HPC36104/HPC46104 High-Performance microControllers with A/D

General Description

The HPC16164 and HPC16104 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC16164 has 16k bytes of on-chip ROM. The HPC16104 has no on-chip ROM and is intended for use with external memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC16164" is used throughout this datasheet to refer to the HPC16164 and HPC16104 devices unless otherwise specified.

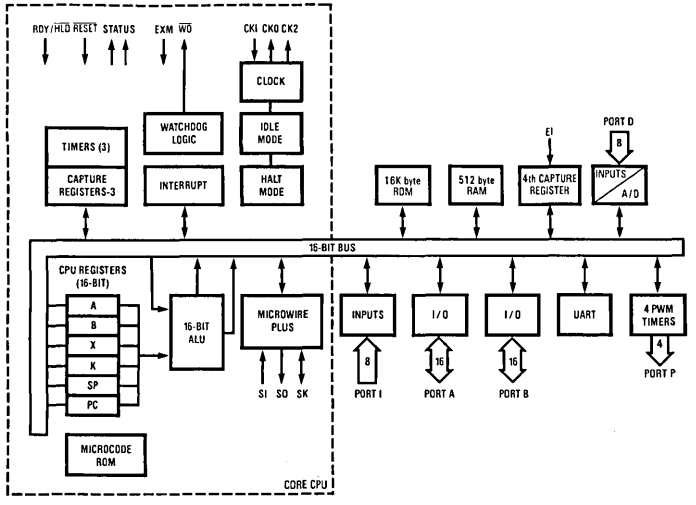
The HPC16164 has, as an on-board peripheral, an 8-channel 8-bit Analog-to-Digital Converter. This A/D converter can operate in single-ended mode where the analog input voltage is applied across one of the eight input channels (D0–D7) and AGND. The A/D converter can also operate in differential mode where the analog input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel pairs in differential mode.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LCC, LDCC, PGA and TapePak™ packages.

Features

- HPC family—core features:
  - 16-bit architecture, both byte and word
  - 16-bit data bus, ALU, and registers
  - 64k bytes of external memory addressing
  - FAST—200 ns for fastest instruction when using 20.0 MHz clock
  - High code efficiency—most instructions are single byte
  - 16 x 16 multiply and 32 x 16 divide
  - Eight vectored interrupt sources
  - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
  - MICROWIRE/PLUS serial I/O interface
  - CMOS—very low power with two power save modes: IDLE and HALT
- A/D—8-channel 8-bit analog-to-digital converter with conversion time minimum 6.6 μs for single conversion
- A/D—supports conversions in "quiet mode"

Block Diagram (HPC16164 with 16k ROM shown)



TL/DD/9682-1

Features (Continued)

- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 16k bytes of ROM, 512 bytes of RAM on-chip
- ROMless version available (HPC16104)
- Commercial (0°C to +70°C), industrial (−40°C to +85°C), automotive (−40°C to +105°C) and military (−55°C to +125°C) temperature ranges

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD Rating	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, −40°C to +85°C for HPC36164/HPC36104, −40°C to +105°C for HPC26164/HPC26104, −55°C to +125°C for HPC16164/HPC16104

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz (Note 1)		60	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		6	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz, (Note 1)		6	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		0.6	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		300	µA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		150	µA
INPUT VOLTAGE LEVELS RESET, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	µA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 µA	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 µA		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA, V_{CC} = 5.0V	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH_4}	ST1 and ST2 Drive	I_{OH} = −6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE® Leakage Current			±5	µA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{RESET}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC} and A/D inactive. CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

AC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, −40°C to +85°C for HPC36164/HPC36104, −40°C to +105°C for HPC26164/HPC26104, −55°C to +125°C for HPC16164/HPC16104

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	20	MHz
$t_{C1} = 1/f_C$	Clock Period	50		ns
$t_C = 2/f_C$	Timing Cycle	100		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns
t_{DC1C2R}	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F}	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		55	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		55	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	18		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	20		ns
$t_{WAIT} = t_C = WS$	Wait State Period	100		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.052	MHz
t_{XIN}	Pulse Width for Timer Inputs	40		ns
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz

Read Cycle Timing with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{RD}$ Falling Edge	20		ns
$t_{RW} = \frac{1}{2} t_C + WS - 10$	$\overline{RD}$ Pulse Width	140		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{RD}$	0	60	ns
$t_{ACC} = t_C + WS - 55$	Address Valid to Input Data Valid		145	ns
$t_{RD} = \frac{1}{2} t_C + WS - 65$	$\overline{RD}$ Falling Edge to Input Data Valid		85	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	95		ns

Write Cycle Timing with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{WR}$ Falling Edge	45		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{WR}$ Pulse Width	160		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{WR}$	20		ns
$t_V = \frac{1}{2} t_C + WS - 15$	Data Valid before Rising Edge of $\overline{WR}$	135		ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 pF, other Outputs C_L = 80 pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC Specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurement will be skewed.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO ***should not*** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a very high confidence level.

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = 1/4\ t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns
$t_{RWP} = t_C$	RDY Pulse Width	100		ns
$t_{SALE} = 3/4\ t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	115		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	110		ns
$t_{HAD} = 7/4\ t_C + 50$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		225	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		200*	ns
t_{BF}	Bus Float before Falling Edge on $\overline{HLDA}$	0		ns
$t_{BE} = 3/4\ t_C + 50$	Bus Enable from Rising Edge of $\overline{HLD}$		125	ns

***Note:** t_{HAE} may be as long as ($3t_C + 4ws + 72t_C + 90$) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value is for the optimal case.

UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{URD}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{URD}$	10		ns
t_{RPW}	$\overline{URD}$ Pulse Width	100		ns
t_{OE}	$\overline{URD}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{URD}$ to Output Data Valid	5	35	ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Rising Edge of $\overline{URD}$		70	ns
t_{WDW}	$\overline{UWR}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{UWR}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{UWR}$	15		ns
t_A	$\overline{WRRDY}$ Delay from Rising Edge of $\overline{UWR}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

A/D Converter Specifications V_{CC} = 5V ±10%

Symbol	Parameter	Min	Max	Units
	Resolution		8	bits
f_{CCLK}	Clock Frequency (Note 4)	0.1	1.6	MHz
$t_{CON} = 10.5/f_{CCLK}$	Conversion Time (Note 3)	6.6		μs
V_{REF}	Reference Voltage Input (AGND = 0V)	3.0	V_{CC}	V
	Total Unadjusted Error (Note 1) (V_{REF} = 5.000V)		±½	LSB
R_{VREF}	Reference Input Resistance	1.6	4.8	kΩ
	DC Common Mode Error		±¼	LSB
	Power Supply Sensitivity (V_{CC} = V_{REF} = 5V ±10%)		±¼	LSB
	Voltage Reference Tolerance (V_{REF})		TBD	LSB
	Analog Input Capacitance		25	pF
	Analog Input Voltage Range (Note 2)	V_{SS} − 0.05	V_{CC} + 0.05	V
	On Channel Leakage		1	μA
	Off Channel Leakage		1	μA

Note 1: Total unadjusted error includes offset, full-scale, and multiplexer errors.

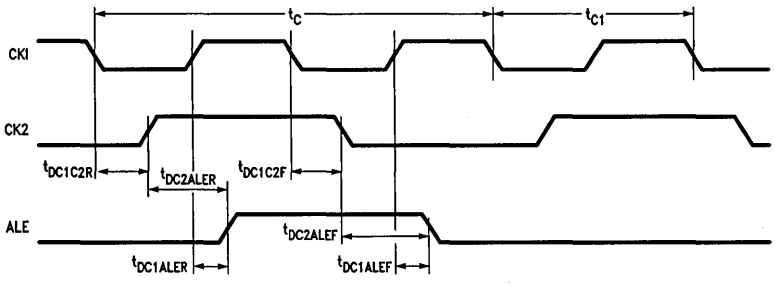
Note 2: 8 single-ended or 4 differential channels. Inherent sample and hold for single-ended inputs (V_{SS} = Pin 62).

Note 3: Conversion time does not include sample/hold time.

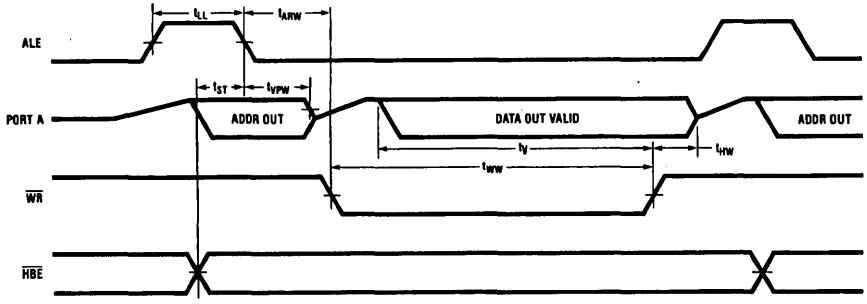
Note 4: Clock supplied to A/D converter is derived from CKI.

Timing Waveforms

CKI, CK2, ALE Timing Diagram

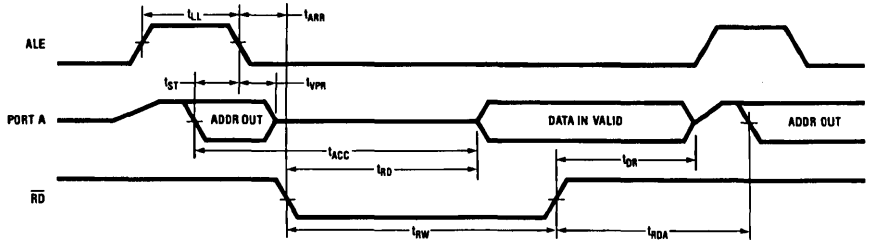


TL/DD/9682-2



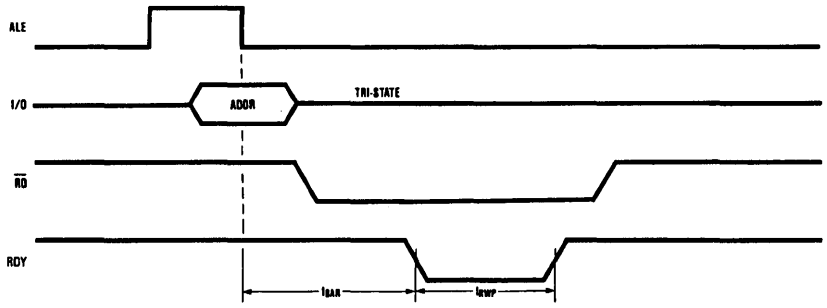
TL/DD/9682-3

FIGURE 1. Write Cycle



TL/DD/9682-4

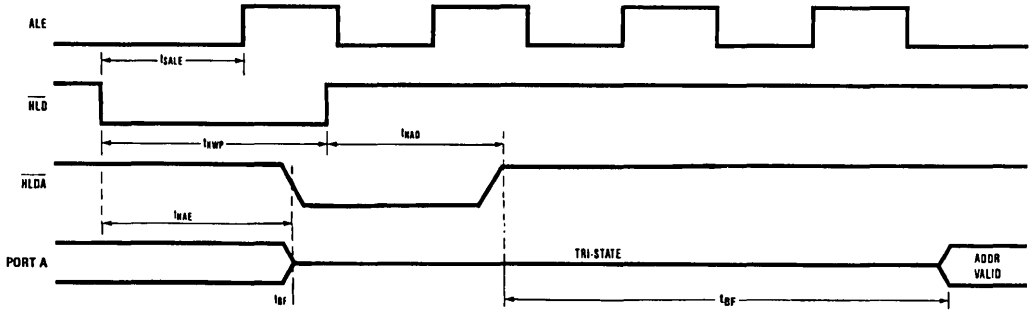
FIGURE 2. Read Cycle



TL/DD/9682-5

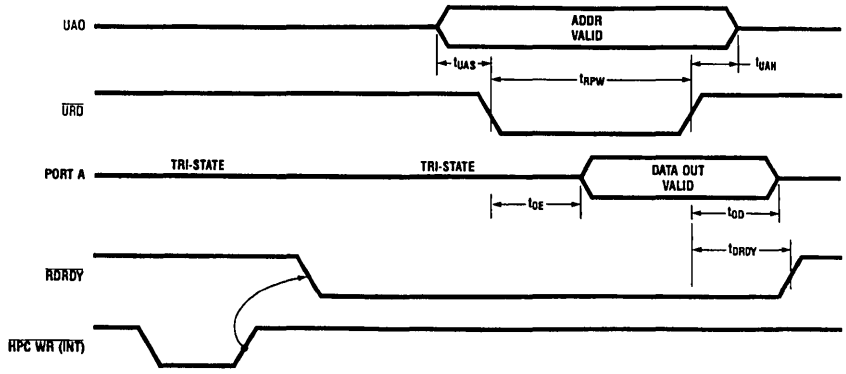
FIGURE 3. Ready Mode Timing

Timing Waveforms (Continued)



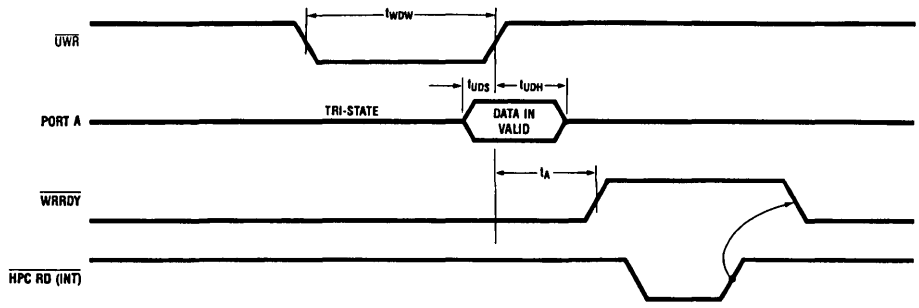
TL/DD/9682-6

FIGURE 4. Hold Mode Timing



TL/DD/9682-9

FIGURE 5. UPI Read Timing



TL/DD/9682-10

FIGURE 6. UPI Write Timing

Pin Descriptions

The HPC16164 is available in 68-pin PLCC, LCC, LDCC, PGA, and TapePak packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{HLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	$\overline{\text{WRRDY}}$	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RDRDY}}$	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	$\overline{\text{WR}}$	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output/Input (sampled at reset)
B15:	$\overline{\text{RD}}$	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/$\overline{\text{URD}}$
I3:	INT3	Maskable Interrupt/Input Capture/$\overline{\text{UWR}}$
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs or as analog channel inputs for the A/D converter. These functions of Port D are mutually exclusive and under the control of software.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V_{CC1} and V_{CC2}	Positive Power Supply (3V to 5.5V)
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI	The Chip System Clock Input
CKO	The Chip System Clock Output (inversion of CKI)

Pins CKI and CKO are usually connected across an external crystal.

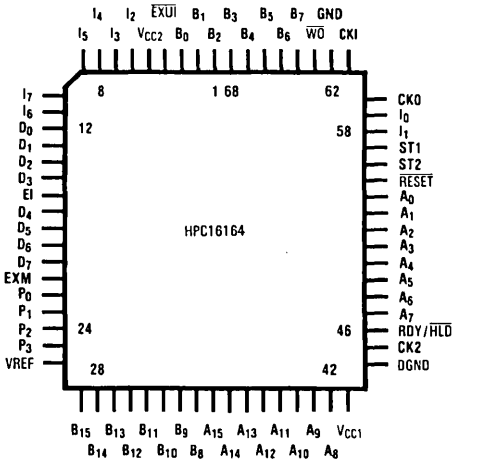
CK2	Clock Output (CKI divided by 2)

OTHER PINS

$\overline{\text{WO}}$	This is an active low open drain output that signals an illegal situation has been detected by the Watch Dog logic.
ST1	Bus Cycle Status Output: indicates first opcode fetch.
ST2	Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
$\overline{\text{RESET}}$	is an active low input that forces the chip to restart and sets the ports in a TRI-STATE® mode.
RDY/$\overline{\text{HLD}}$	has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.
VREF	A/D converter reference voltage input.
EXM	External memory enable (active high) disables internal ROM and maps it to external memory.
EI	External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.
AGND/$\overline{\text{EXUI}}$	has two uses, selected by a software bit. It can be an external active low interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 or it can be the analog ground for the A/D converter.

Connection Diagrams

Plastic, Leadless and Leaded Chip Carriers

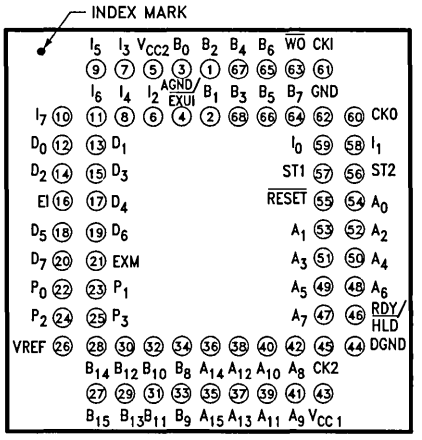


TL/DD/9682-11

Top View

Order Number HPC16164E or V
See NS Package Number E68B or V68A

Pin Grid Array Pinout



TL/DD/9682-12

Top View
(looking down on component side of PC Board)

Order Number HPC16164U
See NS Package Number U68A

4

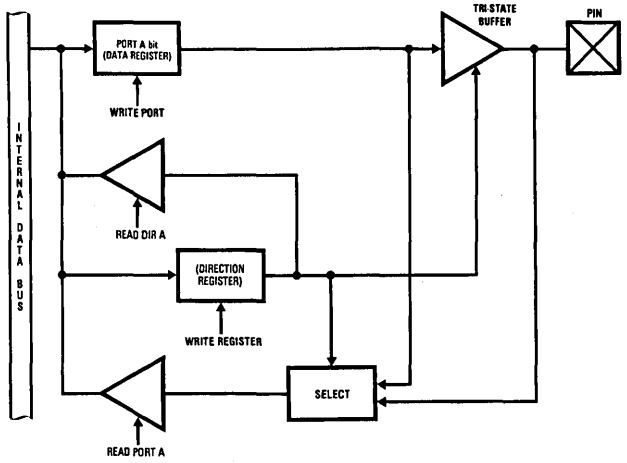
Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 7*), consists of a data register and a direction register. Port B (see *Figures 8, 9* and *10*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

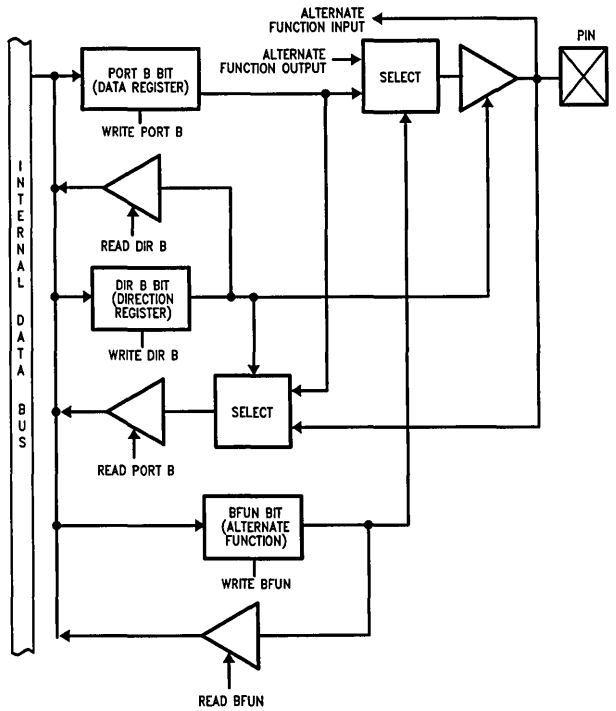
A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.



TL/DD/9682–13

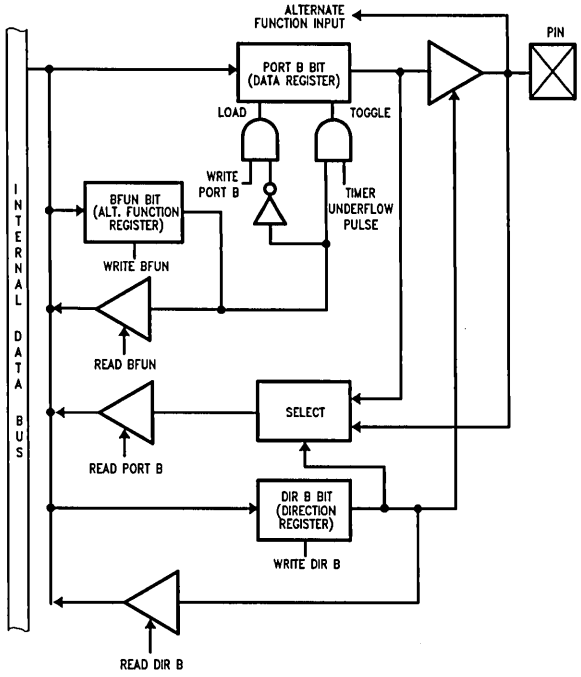
FIGURE 7. Port A: I/O Structure



TL/DD/9682–14

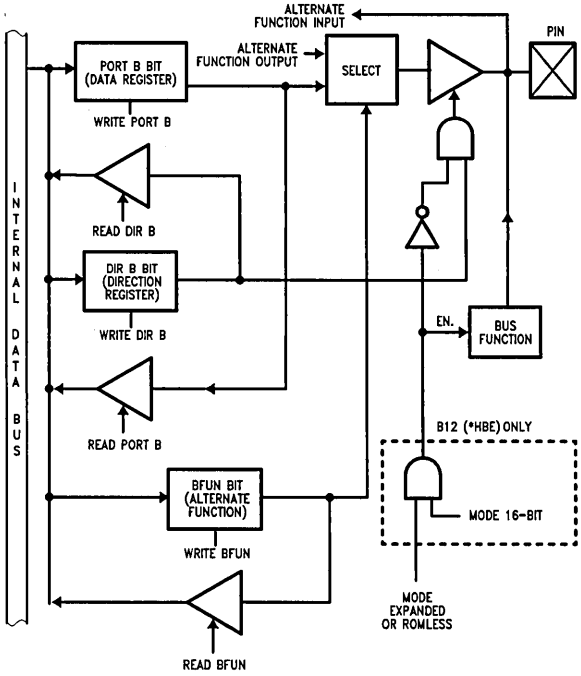
FIGURE 8. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

Ports A & B (Continued)



TL/DD/9682-15

FIGURE 9. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)



TL/DD/9682-16

FIGURE 10. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC16164 and HPC16104 have four operating modes. The ROMless HPC16104 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC16164 is C000 to FFFF (16k bytes). The HPC16104 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC16104 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range and on-chip RAM and Register range, and the "illegal address detection" feature of the Watchdog logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the Watchdog logic is disabled. The EA bit should be set to "1" by software when using the HPC16104 to disable the "illegal address detection" feature of Watchdog.

All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, $\overline{RD}$, $\overline{WR}$ and $\overline{HBE}$. The High Byte Enable pin ($\overline{HBE}$) is used in 16-bit mode to select high order memory bytes. The $\overline{RD}$ and $\overline{WR}$ signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling $\overline{HBE}$ high at reset. If $\overline{HBE}$ is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC16164 and HPC16104.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC16164 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC16164 functions as a self-contained microcomputer (see *Figure 11*) with all memory (RAM and ROM) on-chip. It can address internal memory only, consisting of 16k bytes of ROM (C000 to FFFF) and 512 bytes of on-chip RAM and Registers (0000 to 02FF). The "illegal address detection" feature of the Watchdog is enabled in the Single-Chip Normal mode and a Watchdog Output ($\overline{WO}$) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC16164 to address external memory in addition to the on-chip ROM and RAM (see Table II). Watchdog illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC16164 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 16k of external memory may be used with the HPC16164 (see Table II). The Watchdog circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

EXPANDED ROMLESS MODE

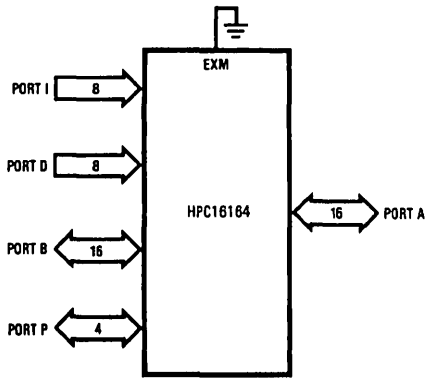
This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of Watchdog is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

TABLE II. HPC16164 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	C000:FFFF on-chip
Expanded Normal	0	1	C000:FFFF on-chip 0300:BFFF off-chip
Single-Chip ROMless	1	0	C000:FFFF off-chip
Expanded ROMless	1	1	0300:FFFF off-chip

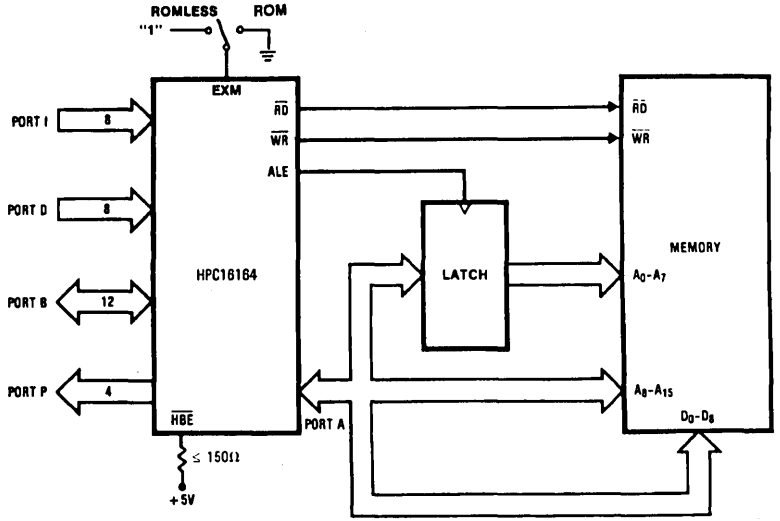
Note: In all operating modes, the on-chip RAM and Registers (0000:02FF) may be accessed.

HPC16164 Operating Modes (Continued)



TL/DD/9682-17

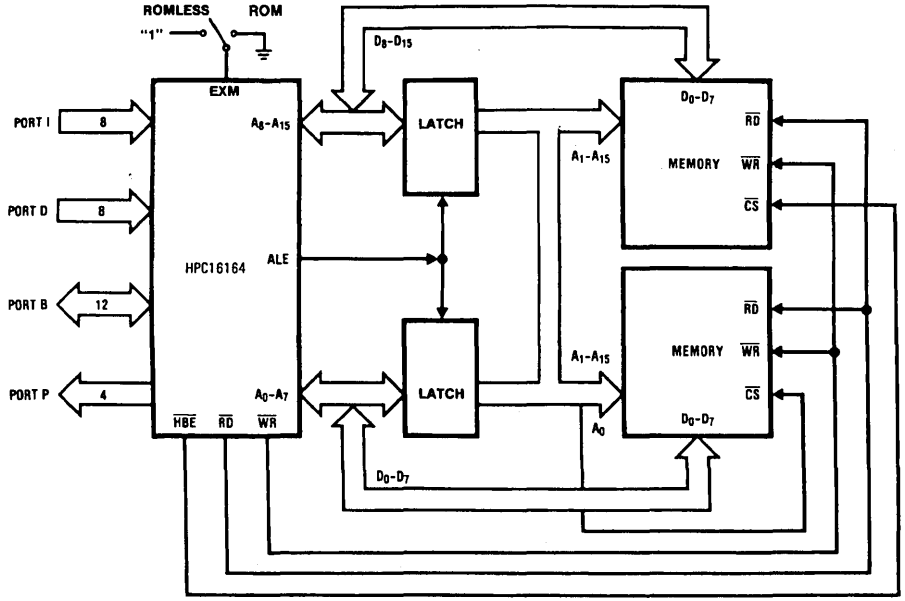
FIGURE 11. Single-Chip Mode



TL/DD/9682-18

FIGURE 12. 8-Bit External Memory

HPC16164 Operating Modes (Continued)



TL/DD/9682-19

FIGURE 13. 16-Bit External Memory

HPC16104 Operating Modes

EXPANDED ROMLESS MODE (HPC16104)

Because the HPC16104 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC16104 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between C000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the Watchdog logic.

TABLE III. HPC16104 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0300:FFFF off-chip

Note: The on-chip RAM and Registers (0000:02FF) of the HPC16104 may be accessed at all times.

Wait States

The HPC16164 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC16164: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16164 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16164 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the $\overline{\text{RESET}}$ or the NMI. The $\overline{\text{RESET}}$ input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16164 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT

Power Save Modes (Continued)

mode, the processor is returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC16164 to resume normal operation.

HPC16164 Interrupts

Complex interrupt handling is easily accomplished by the HPC16164's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table IV.

TABLE IV. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
$FFFF:FFFE	RESET	0
$FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
$FFFB:FFFA	External interrupt on I2 pin	2
$FFF9:FFF8	External interrupt on I3 pin	3
$FFF7:FFF6	External interrupt on I4 pin	4
$FFF5:FFF4	Overflow on internal timers	5
$FFF3:FFF2	Internal by on-board peripherals or external on EXUI	6
$FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC16164 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table IV. The interrupt on Reset has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately. RESET and EXUI are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt (EXUI) is shared with the on-board peripherals, UART and A/D. The EXUI interrupt is level-LOW-sensitive. To select this interrupt, disable the ERI and ETI UART interrupts by resetting these enable bits in the ENUI register and disable the A/D function by resetting the ADEN bit in the A/D control register #3 (CR3). To select the on-board peripherals interrupt, leave this pin floating or tie it high if the A/D function is disabled. If the A/D function is enabled, this pin becomes the analog ground (AGND).

Interrupt Control Registers

The HPC16164 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to be serviced, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16164 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 14* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A and B in the TRI-STATE condition and Port P in the LOW state. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between C000 and FFFF when using the HPC16104.

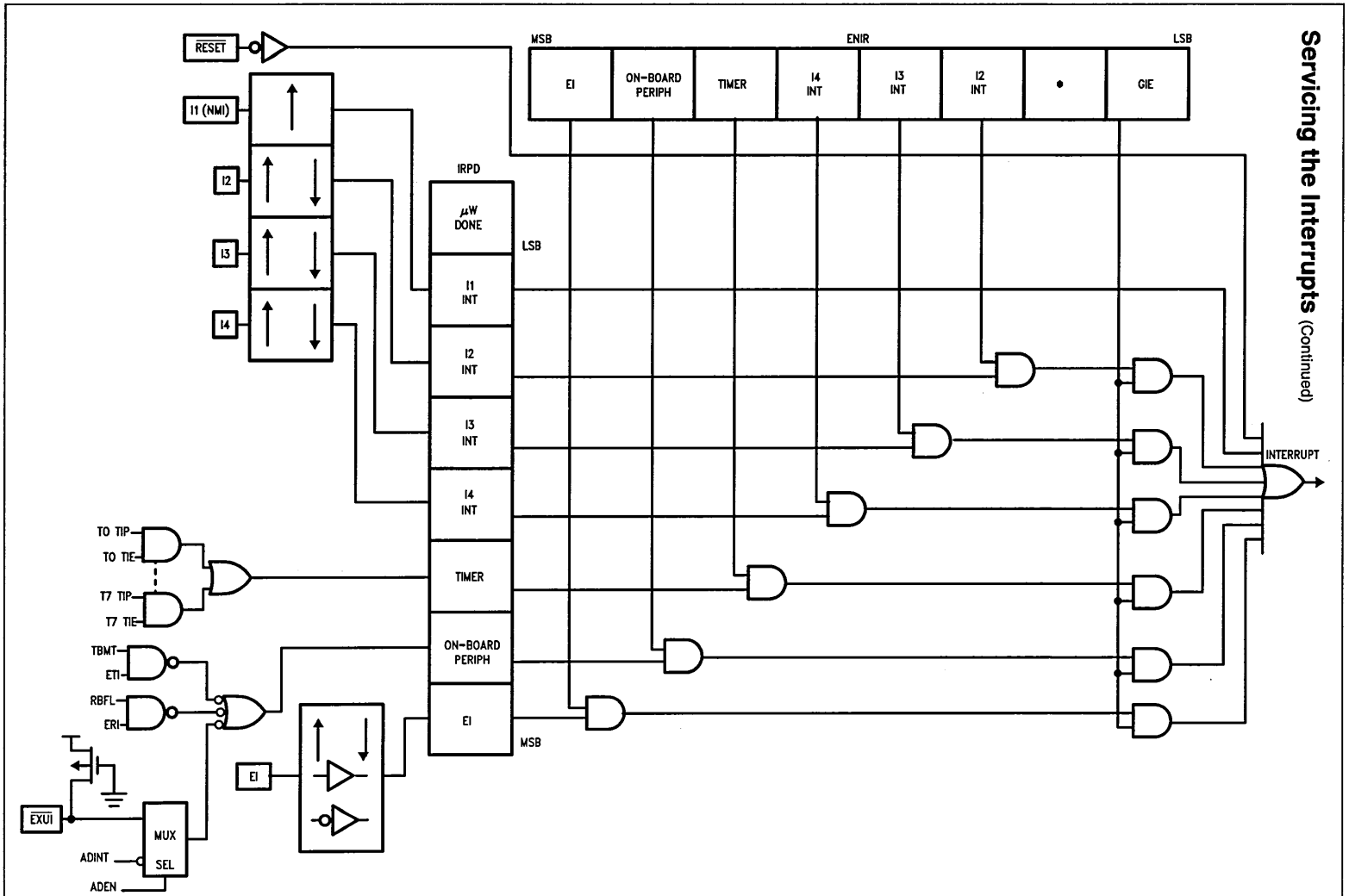


FIGURE 14. Block Diagram of Interrupt Logic

Timer Overview

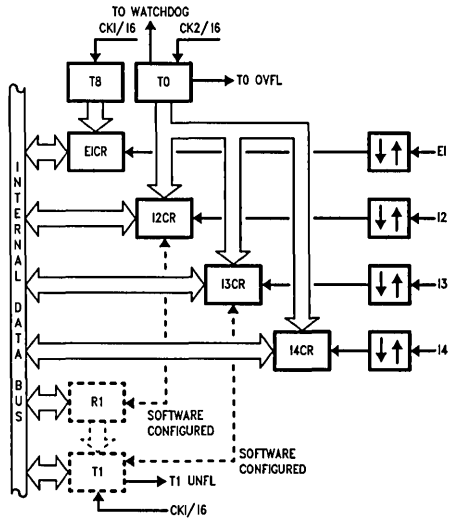
The HPC16164 contains a powerful set of flexible timers enabling the HPC16164 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16164 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watchdog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 15*).

The HPC16164 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 16*).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

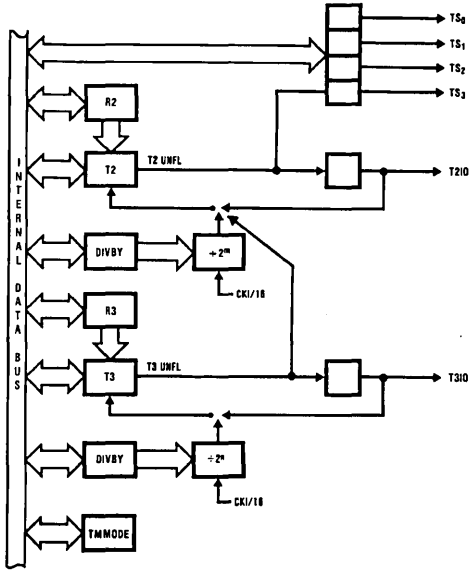


TL/DD/9682-21

FIGURE 15. Timers T0, T1 and T8 with Four Input Capture Registers

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16164 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 16*).

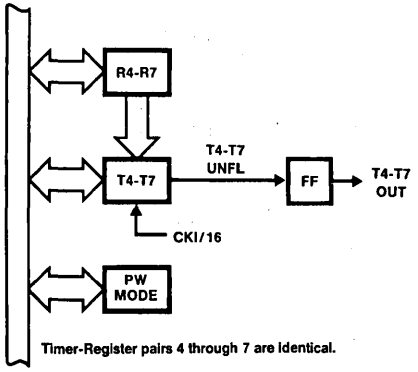


TL/DD/9682-22

FIGURE 16. Timers T2–T3 Block

Timer Overview (Continued)

Timer/register pairs 4–7 form four identical units which can generate synchronous outputs on port P (see *Figure 17*). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to ½ the frequency of the source used for clocking the timer.



TL/DD/9682–23

FIGURE 17. Timers T4–T7 Block

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16164.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

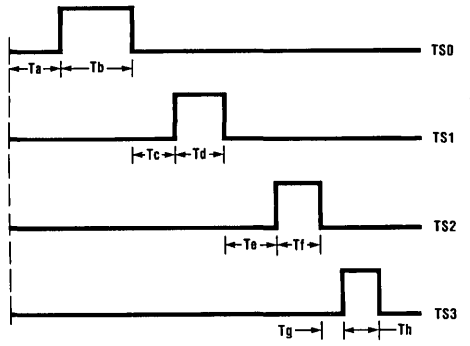
TL/DD/9682–24

FIGURE 18. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 19* is an example of synchronous pulse train generation.

Watchdog Logic

The Watchdog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watchdog logic are potentially infinite loops and illegal addresses. Should the Watchdog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the Watchdog Output ($\overline{WO}$) pin low. The $\overline{WO}$ pin is an open drain output and can be connected to the $\overline{RESET}$ or NMI inputs or to the users external logic.



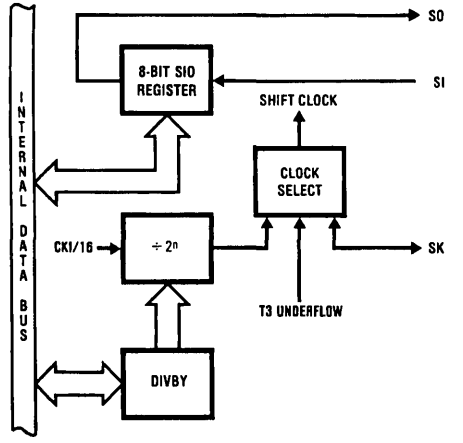
TL/DD/9682–25

FIGURE 19. Synchronous Pulse Generation

***Note:** See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 20*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.



TL/DD/9682–26

FIGURE 20. MICROWIRE/PLUS

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS Operation

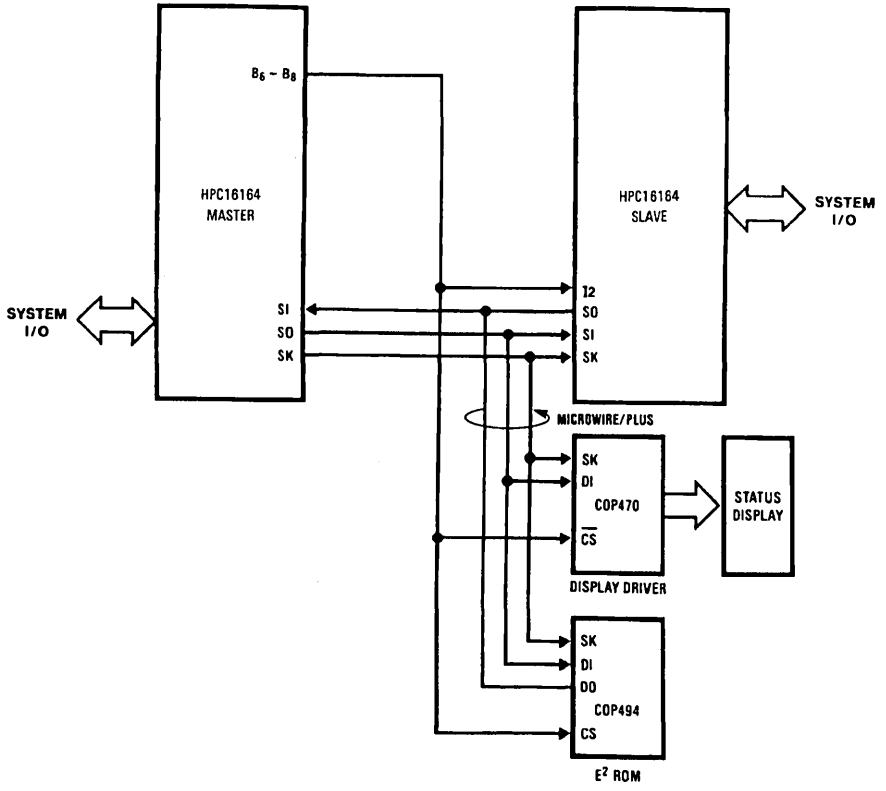
The HPC16164 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16164 is the master or slave. The shift clock is generated when the HPC16164 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16164 is configured as a slave. When the HPC16164 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 21 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC16164 microcontrollers interconnected to other MICROWIRE peripherals. HPC16164 #1 is set up as the master and initiates all data transfers. HPC16164 #2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of a VF display controlled by the COP470 display driver. The data to be displayed is sent serially to the COP470 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC16164 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.



TL/DD/9682-27

FIGURE 21. MICROWIRE/PLUS Application

HPC16164 UART

The HPC16164 contains a software programmable UART. The UART (see *Figure 22*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

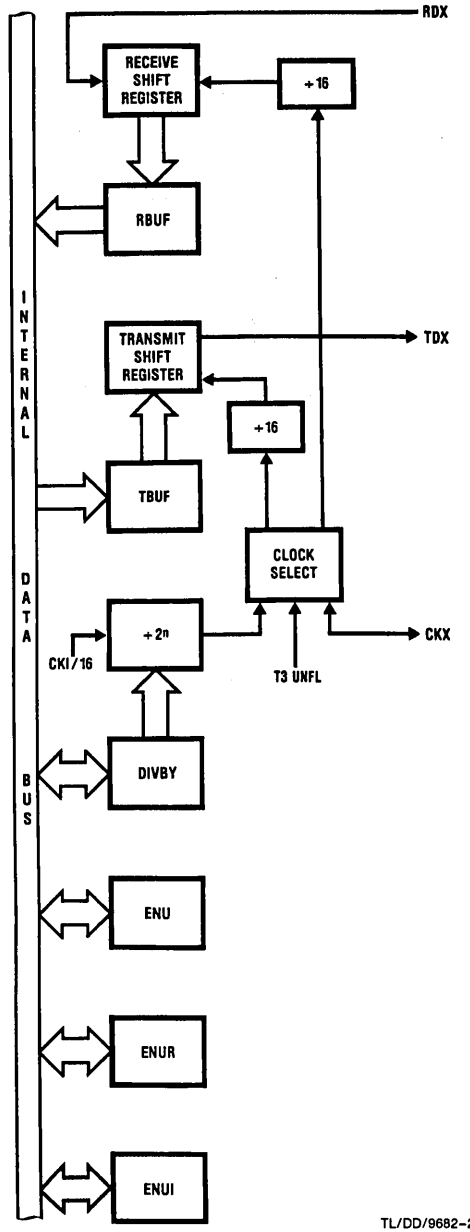
The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC16164 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16164 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16164 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16164 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.



TL/DD/9682-28

FIGURE 22. UART Block Diagram

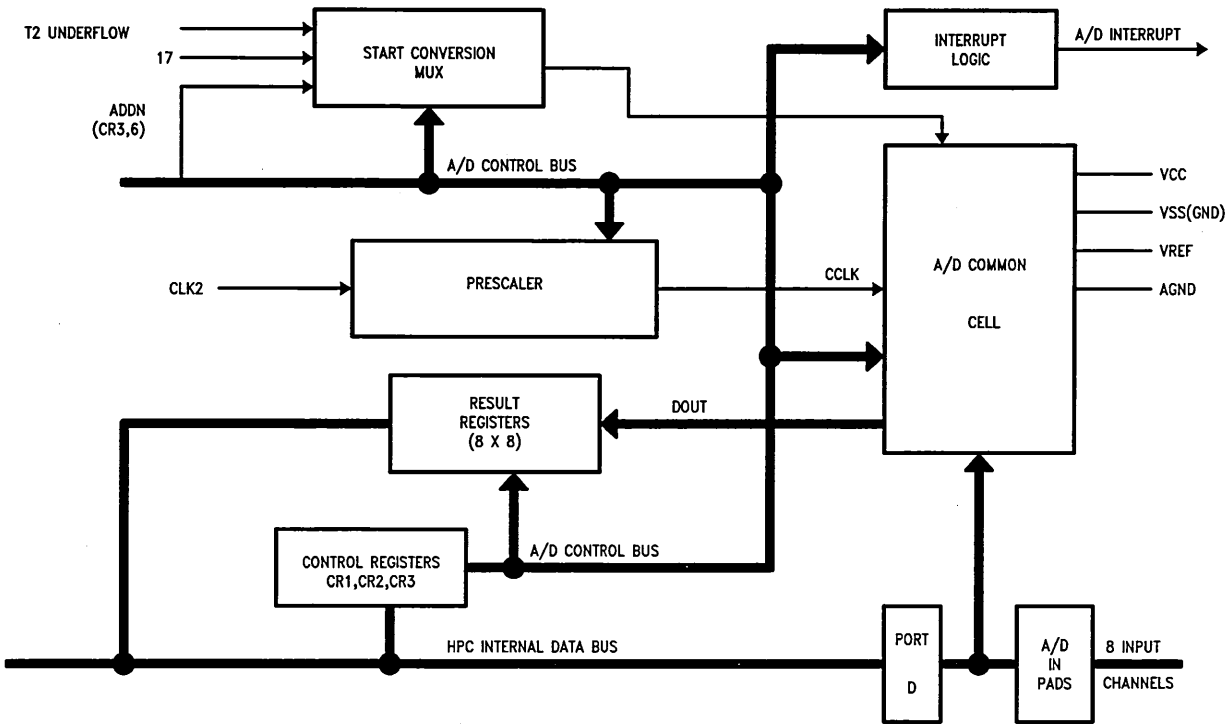


FIGURE 23. A/D Block Diagram

A/D Converter Operation (Continued)

The HPC16164 has an on-board eight-channel 8-bit Analog to Digital converter. The A/D converter cell can operate in single-ended mode where the input voltage is applied across one of the eight input channels (D0–D7) and AGND or in differential mode where the input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel-pairs in differential mode.

OPERATING MODES

The operating modes of the converter are selected by 4 bits called ADMODE (CR2.4–7). Associated with the eight input channels in single-ended mode are eight result registers, one for each channel. The A/D converter can be programmed by software to convert on any specific channel storing the result in the result register associated with that channel. It can also be programmed to stop after one conversion or to convert continuously. If a brief history of the signal on any specific input channel is required, the converter can be programmed to convert on that channel and store the consecutive results in each of the result registers before stopping. As a final configuration in single-ended mode, the converter can be programmed to convert the signal on each input channel and store the result in its associated result register continuously.

Associated with each even-odd pair of input channels in differential mode of operation are four result register-pairs. The A/D converter performs two conversions on the selected pair of input channels. One conversion is performed assuming the positive connection is made to the even channel and the negative connection is made to the following odd channel. This result is stored in the result register associated with the even channel. Another conversion is performed assuming the positive connection is made to the odd channel and the negative connection is made to the preceding even channel. This result is stored in the results register associated with the odd channel. This technique does not require that the programmer know the polarity of the input signal. If the even channel result register is non-zero (meaning the odd channel result register is zero), then the input signal is positive with respect to the odd channel. If the odd channel result register is non-zero (meaning the even channel result register is zero), then the input signal is positive with respect to the even channel.

The same operating modes for single-ended operation also apply when the inputs are taken from channel-pairs in differential mode. The programmer can configure the A/D to convert on any selected channel-pair and store the result in its associated result register-pair then stop. The A/D can also be programmed to do this continuously. Conversion can also be done any channel-pair storing the result into four result register-pairs for a history of the differential input. Finally, all input channel-pairs can be converted continuously.

The final mode of operation suppresses the external address/data bus activity during the single conversion modes. These quiet modes of operation utilize the RDY function of the HPC Core to insert wait states in the instruction being executed in order to limit digital noise in the environment due to external bus activity when addressing external memory. The overall effect is to increase the accuracy of the A/D.

CONTROL

The conversion clock supplied to the A/D converter can be selected by three bits in CR1 used as a prescaler on CKI. These bits can be used to ensure that the A/D is clocked as fast as possible when different external crystal frequencies are used. Controlling the starting of conversion cycles in each of the operating modes can be done by four different methods. The method is selected by two bits called SC (CR3.0–1). Conversion cycles can be initiated through software by resetting a bit in a control register, through hardware by an underflow of Timer T2, or externally by a rising or falling edge of a signal input on I7.

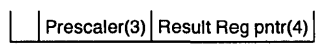
INTERRUPTS

The A/D converter can interrupt the HPC when it completes a conversion cycle if one of the non-continuous modes has been selected. If one of the cycle modes was selected, then the converter will request an interrupt after eight conversions. If one of the one-shot modes was selected, then the converter will request an interrupt after every conversion. The A/D converter does not have its own interrupt vector location. When this interrupt is generated, the HPC vectors to the on-board peripheral interrupt vector location at address FFF2. The service routine must then determine if the A/D converter requested the interrupt by checking the A/D done flag which doubles as the A/D interrupt pending flag.

REGISTER MAP

The A/D converter status and control registers and the result registers are detailed as follows:

Control Register #1 (CR1)



byte at location 0100

Result Register pointer—These four bits are read/only by the software. In all the operating modes that are single channel or single channel-pair, this pointer gets the value of the Channel Select bits (CR2.0–3) and remains constant. In the operating modes that work on multiple channels or multiple channel-pairs, this pointer gets initialized to zero and will change to reflect the current channel that is being converted (default value on power-up is 0000).

Prescaler—These three bits are used to select the clock (CCLK) supplied to the SAR in the A/D converter cell. The maximum clock that can be supplied is 1.67 MHz and the minimum is 100 kHz. Therefore, these bits can be used to ensure that the A/D is clocked as fast as posible at different external crystal frequencies.

000 = stop the clock (CCLK) to the A/D cell (default value on power-up)

011 = use CKI/4 to allow max CKI of 6.66 MHz

010 = use CKI/8 to allow max CKI of 13.33 MHz

111 = use CKI/12 to allow max CKI of 20 MHz

101 = use CKI/16 to allow max CKI of 26.66 MHz

001 = use CKI/20 to allow max CKI of 33.33 MHz

110 = use CKI/24 to allow max CKI of 40 MHz

100 = use CKI/32 to allow max CKI of 53.4 MHz

Note: All remaining unused bits in this control register are UNDEFINED and not available for use by the program.

A/D Converter Operation (Continued)

Control Register #2 (CR2)

ADMODE(4)	Channel Select(4)

msb lsb

byte at location 0102

ADMODE—These four bits are used to select the mode of operation for the A/D converter as described in OPERATING MODES.

0000 = single-ended, single channel, single result register, one-shot (default value on power-up)

0001 = single-ended, single channel, single result register, continuous

0010 = single-ended, single channel, multiple result registers, stop after 8

0011 = single-ended, multiple channel, multiple result registers, continuous

0100 = differential, single channel-pair, single result register-pair, one-shot

0101 = differential, single channel-pair, single result register-pair, continuous

0110 = differential, single channel-pair, multiple result register-pairs, stop after 4 pairs

0111 = differential, multiple channel-pair, multiple result register-pairs, continuous

Channel Select—These four bits are used to select the channel on which to initiate conversions.

Single-ended

x000 = Convert on Channel 0 (Input Port D.0)

x001 = Convert on Channel 1 (Input Port D.1)

x010 = Convert on Channel 2 (Input Port D.2)

x011 = Convert on Channel 3 (Input Port D.3)

x100 = Convert on Channel 4 (Input Port D.4)

x101 = Convert on Channel 5 (Input Port D.5)

x110 = Convert on Channel 6 (Input Port D.6)

x111 = Convert on Channel 7 (Input Port D.7)

Differential

x000 = Convert on Channel-Pair 0,1

x010 = Convert on Channel-Pair 2,3

x100 = Convert on Channel-Pair 4,5

x110 = Convert on Channel-Pair 6,7

Control Register #3 (CR3)

	ADDN			ADIE	ADEN	SC Mode (2)

msb lsb

byte at location 0106

SC mode—These two bits are used to select the mode for starting a conversion cycle.

00 = A conversion cycle is initiated by resetting the A/D done flag (ADDN) (default value on power-up).

01 = A conversion cycle is initiated by an underflow of Timer T2.

10 = A conversion cycle is initiated by the falling edge of the signal on input I7.

11 = A conversion cycle is initiated by the rising edge of the signal on input I7.

ADEN—Setting this bit enables pin 4 to be the analog ground, AGND. Resetting this bit returns pin 4 as $\overline{\text{EXUI}}$ (reset on power-up).

ADIE—This is the A/D interrupt enable bit. (reset on power-up).

ADDN—This bit is the A/D done flag and doubles as the A/D interrupt pending flag. If one of the one-shot modes was selected using ADMODE(=xx00) and control was selected as SC = 00, then this bit must be reset by software to initiate the conversion and is set by the hardware at the end of one conversion. If one of the cycle modes was selected using ADMODE(=xx10) and control was selected as SC = 00, then this bit must be reset by software to initiate the conversion cycle and is not set by the hardware until the end of one conversion cycle. If any of the continuous modes were selected and control was selected as SC = 00, then this bit must be reset by software to initiate the conversions and is not set by the hardware until the clock to the A/D cell is stopped by selecting the value 000 for the prescaler. In all other control selections, this bit has no effect on the initiation of conversions but is still necessary for proper interrupt operation. The ADDN flag must also be reset for the quiet modes to work properly (set on power-up).

Note: All remaining unused bits in this control register are UNDEFINED and not available for use by the program. Also, all result register contents are UNDEFINED on power-up.

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC16164 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC16164's and set up systems with very high data exchange rates. Another area of application could be where a HPC16164 is programmed as an intelligent peripheral to a host system such as the Series 32000® microprocessor. *Figure 24* illustrates how a HPC16164 could be used as an intelligent peripherial for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe ($\overline{\text{URD}}$), a Write Strobe ($\overline{\text{UWR}}$), a Read Ready Line ($\overline{\text{RDRDY}}$), a Write Ready Line ($\overline{\text{WRRDY}}$) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The $\overline{\text{URD}}$ and $\overline{\text{UWR}}$ inputs may be used to interrupt the HPC16164. The $\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$ outputs may be used to interrupt the host processor.

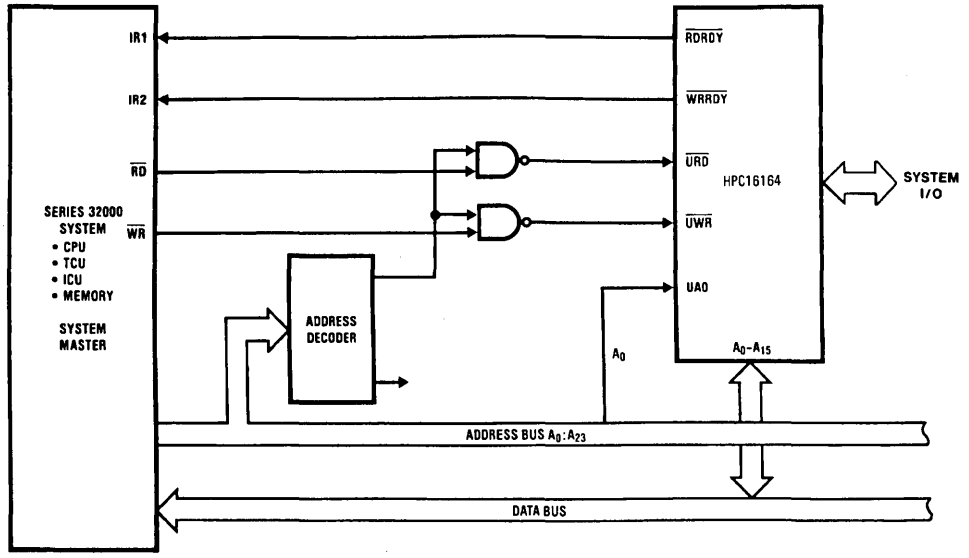
The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC16164 is the data bus. UPI can only be used if the HPC16164 is in the Single-Chip mode.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16164 supports shared memory access with two pins. The pins are the RDY/$\overline{HLD}$ input pin and the $\overline{HLDA}$ output pin. The user can software select either the Hold or Ready function by the state of a control bit. The $\overline{HLDA}$ output is multiplexed onto port B.

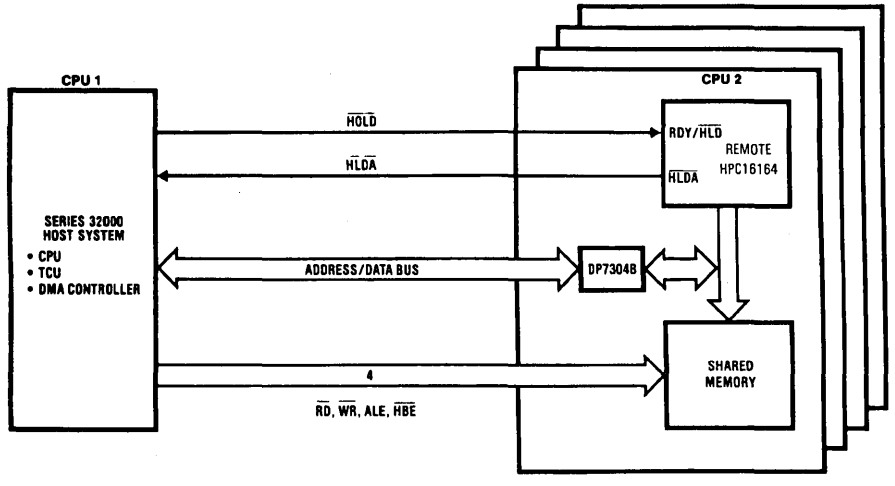
The host uses DMA to interface with the HPC16164. The host initiates a data transfer by activating the $\overline{HLD}$ input of the HPC16164. In response, the HPC16164 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{HLDA}$) from the HPC16164 indicating that the sytem bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16164 resumes normal operations.

Figure 25 illustrates an application of the shared memory interface between the HPC16164 and a Series 32000 system.



TL/DD/9682-30

FIGURE 24. HPC16164 as a Peripheral: (UPI Interface to Series 32000 Application)



TL/DD/9682-31

FIGURE 25. Shared Memory Application: HPC16164 Interface to Series 32000 System

Memory

The HPC16164 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 16 kbytes of ROM and 512 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16164 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC16164 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table V.

TABLE V. HPC16164 Memory Map

FFFF:FFF0	Interrupt Vectors	
FFEF:FFD0	JSRP Vectors	
FFCF:FFCE		
: :	On-Chip ROM*	
E001:C000		USER MEMORY
BFFF:BFFE		
: :	External Expansion	
0301:0300	Memory	
02FF:02FE		
: :	On-Chip RAM	USER RAM
01C1:01C0		
0195:0194	Watchdog Address	Watchdog Logic
0192	T0CON Register	
0191:0190	TMMODE Register	
018F:018E	DIVBY Register	
018D:018C	T3 Timer	
018B:018A	R3 Register	
0189:0188	T2 Timer	Timer Block T0:T3
0187:0186	R2 Register	
0185:0184	I2CR Register/ R1	
0183:0182	I3CR Register/ T1	
0181:0180	I4CR Register	
015E:015F	EICR	
015C	EICON	
0153:0152	Port P Register	
0151:0150	PWMODE Register	
014F:014E	R7 Register	
014D:014C	T7 Timer	
014B:014A	R6 Register	Timer Block T4:T7
0149:0148	T6 Timer	
0147:0146	R5 Register	
0145:0144	T5 Timer	
0143:0142	R4 Register	
0141:0140	T4 Timer	
0128	ENUR Register	
0126	TBUF Register	
0124	RBUF Register	UART
0122	ENUI Register	
0120	ENU Register	

011F:011E	A/D Result Register 7	
011D:011C	A/D Result Register 6	
011B:011A	A/D Result Register 5	
0119:0118	A/D Result Register 4	
0117:0116	A/D Result Register 3	
0115:0114	A/D Result Register 2	
0113:0112	A/D Result Register 1	
0111:0110	A/D Result Register 0	
0106	A/D Control Register #3	
0104	Port D / A/D Analog Channel Inputs	
0102	A/D Control Register #2	
0100	A/D Control Register #1	
00F5:00F4	BFUN Register	PORTS A & B CONTROL
00F3:00F2	DIR B Register	
00F1:00F0	DIR A Register / IBUF	
00E6	UPIC Register	UPI CONTROL
00E3:00E2	Port B	PORTS A & B
00E1:00E0	Port A / OBUF	
00DE	Microcode ROM Dump	
00DD:00DC	HALT Enable Register	
00D8	Port I Input Register	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00D6	SIO Register	
00D4	IRCD Register	
00D2	IRPD Register	
00D0	ENIR Register	
00CF:00CE	X Register	
00CD:00CC	B Register	
00CB:00CA	K Register	
00C9:00C8	A Register	HPC CORE REGISTERS
00C7:00C6	PC Register	
00C5:00C4	SP Register	
00C3:00C2	(reserved)	
00C0	PSW Register	
00BF:00BE		
: :	On-Chip RAM	USER RAM
0001:0000		

***Note:** The HPC16164 On-Chip ROM is on addresses C000:FFFF and the External Expansion Memory is 0300:EFFF. The HPC16104 has no On-Chip ROM, External Memory is 0300:FFFF.

HPC16164 CPU

The HPC16164 CPU has a 16-bit ALU and six 16-bit registers

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16164 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	MA + Meml → MA carry → C
ADC	Add with carry	MA + Meml + C → MA carry → C
ADDS	Add short imm8	A + imm8 → A carry → C
DADC	Decimal add with carry	MA + Meml + C → MA (Decimal) carry → C
SUBC	Subtract with carry	MA − Meml + C → MA carry → C
DSUBC	Decimal subtract w/carry	MA − Meml + C → MA (Decimal) carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C
DIVD	Divide Double Word (unsigned)	X & MA/Meml → MA, rem → X, 0 → K, Carry → C
IFEQ	If equal	Compare MA & Meml, Do next if equal
IFGT	If greater than	Compare MA & Meml, Do next if MA > Meml
AND	Logical and	MA and Meml → MA
OR	Logical or	MA or Meml → MA
XOR	Logical exclusive-or	MA xor Meml → MA
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	Mem + 1 → Mem
DECSZ	Decrement, skip if 0	Mem − 1 → Mem, Skip next if Mem = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Meml → MA
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ±1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP
POP	Pop Stack to Memory	SP−2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A,B ± 1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B,imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A − 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP),SP+2 → SP, W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP),SP+2 → SP,PC+ # → PC (#is +1025 to −1023)
JSRL	Jump subroutine long	PC → W(SP),SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +257 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP−2 → SP,W(SP) → PC
RETSK	Return then skip next	SP−2 → SP,W(SP) → PC, & skip
RETI	Return from interrupt	SP−2 → SP,W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory

MA is Accumulator A or direct memory (8 or 16-bit)

Mem is 8-bit byte or 16-bit word of memory

Meml is 8- or 16-bit memory or 8 or 16-bit immediate data

imm is 8-bit or 16-bit immediate data

imm8 is 8-bit immediate data only

4

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir. (B)	Reg Indir. (X)	Direct	Indir	Index	Immed.	Direct *	Direct **	Immed. *	Immed. **
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPS™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBM-R	Relocatable Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001
	424410897-001	Users Manual		424410897-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

Information System

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP
Information System Package contains DIAL-A-HELPER users manual P/N Public Domain Communications Software.

Factory Applications Support

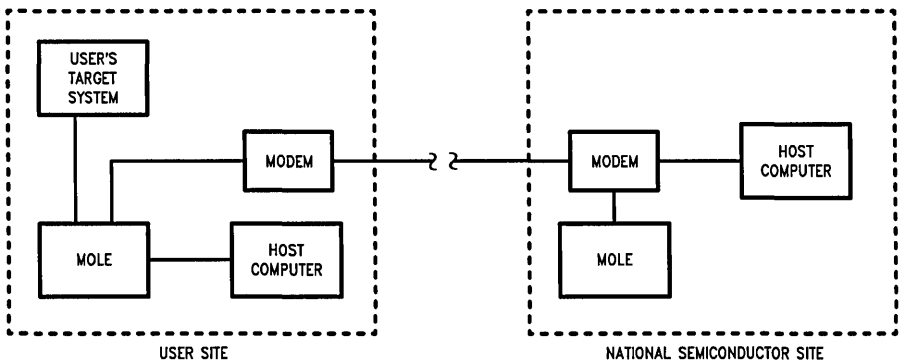
Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occuring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customers problem by actually getting customers system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Setup: Length: 8-Bit
Parity: None
Stop: Bit
Operation: 24 Hrs. 7 Days

DIAL-A-HELPER



TL/DD/9682-32

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16164 has been designed to be extremely code-efficient. The HPC16164 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16164, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16164 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16164 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16164 supplies 8-bit byte capability for 2-digit variables and literal variables.

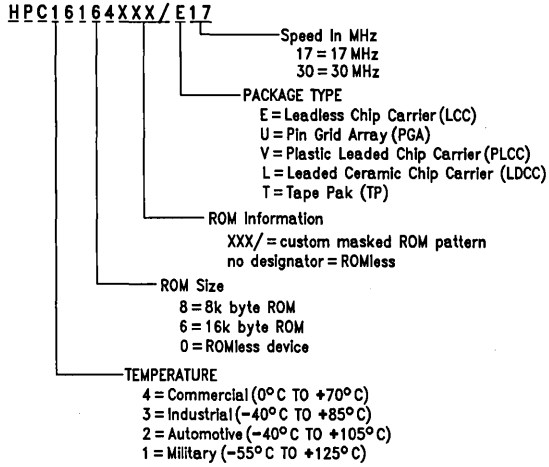
MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16164 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16164 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/9682-33

FIGURE 8. HPC Family Part Numbering Scheme

Examples

HPC46104E17 — ROMless, Commercial temp. (0°C to 70°C), LCC

HPC16164XXX/U17 — 16k masked ROM, Military temp. (−55°C to +125°C), PGA

HPC26104XXX/V17 — ROMless, Automotive temp. (−40°C to +105°C), PLCC

National Semiconductor

ADVANCE INFORMATION

HPC16400/HPC36400/HPC46400 High-Performance microControllers with HDLC Controller

General Description

The HPC16400 is a member of the HPC™ family of High Performance microControllers. Each member of the family has the same identical core CPU with a unique memory and I/O configuration to suit specific applications. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC16400 has 4 functional blocks to support a wide range of communication application-2 HDLC channels, 4 channel DMA controller to facilitate data flow for the HDLC channels, programmable serial interface and UART.

The serial interface decoder allows the 2 HDLC channels to be used with devices using interchip serial link for point-to-point & multipoint data exchanges. The decoder generates enable signals for the HDLC channels allowing multiplexed D and B channel data to be accessed.

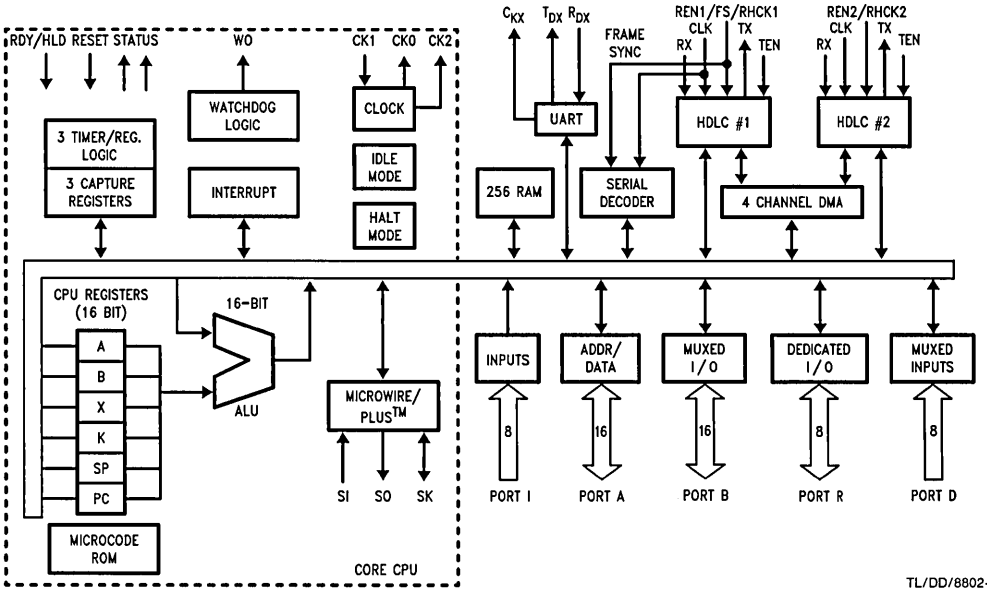
The HDLC channels manage the link by providing sequencing using the HDLC framing along with error control based upon a cyclic redundancy check (CRC). Multiple address recognition modes, and both bit and byte modes of operation are supported.

The HPC16400 is available in 68-pin PLCC, LCC, LDCC and PGA packages.

Features

- HPC family—core features:
  - — 16-bit data bus, ALU, and registers
  - — 64 kbytes of external memory addressing
  - — FAST!—20.0 MHz system clock
  - — High code efficiency
  - — 16 x 16 multiply and 32 x 16 divide
  - — Eight vectored interrupt sources
  - — Four 16-bit timer/counters with WATCHDOG logic
  - — MICROWIRE/PLUS serial I/O interface
  - — CMOS—low power with two power save modes
- Two full duplex HDLC channels
  - — Optimized for X.25 and LAPD applications
  - — Programmable frame address recognition
  - — Up to 4.65 Mbps serial data rate
  - — Built in diagnostics
- Programmable interchip serial data decoder
- Four channel DMA controller
- UART—full duplex, programmable baud rate (up to 208.3 kBaud)
- 544 kbytes of extended addressing
- Easy interface to National's DASL, 'U' and 'S' transceivers—TP3400, TP3410 and TP3420
- Industrial (−40°C to +85°C) and military (−55°C to +125°C) temperature ranges

Block Diagram



TL/DD/8802-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
ESD Rating	2000V
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46400, −40°C to +85°C for HPC36400, −55°C to +125°C for HPC16400

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz* (Note 1)		70	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		7	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz (Note 1)		10	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		1	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz (Note 1)		300	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz (Note 1)		150	μA
INPUT VOLTAGE LEVELS RESET, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
PORT A					
V_{IH_2}	Logic High		2.0		V
V_{IL_2}	Logic Low			0.8	V
ALL OTHER INPUTS					
V_{IH_3}	Logic High		0.7 V_{CC}		V
V_{IL_3}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS CMOS OPERATION					
V_{OH_1}	Logic High	I_{OH} = −10 μA	V_{CC} − 0.1		V
V_{OL_1}	Logic Low	I_{OH} = 10 μA		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA, V_{CC} = 5.0V	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH_4}	ST1 and ST2 Drive	I_{OH} = −6 mA, V_{CC} = 5.0V	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5		V
I_{OZ}	TRI-STATE Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with RESET = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}. CKI driven to V_{IH_1} and V_{IL_1} with rise and fall times less than 10 ns.

Note 2: These parameters are guaranteed by design and are not tested.

Note 3: Test duration is 100 ns.

AC Electrical Characteristics V_{CC} = 5.0V ±10%, f_C = 16.78 MHz, T_A = 0°C to +70°C for HPC46400, −40°C to +85°C for HPC36400, −55°C to +125°C for HPC16400

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2.0	20	MHz
$t_{C1} = 1/f_C$	Clock Period	50		ns
$t_C = 2/f_C$	Timing Cycle	100		ns
$t_{LL} = ½\,t_C - 9$	ALE Pulse Width	41		ns
t_{DC1C2R}	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F}	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = ¼\,t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		55	ns
$t_{DC2ALEF} = ¼\,t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		55	ns
$t_{ST} = ¼\,t_C - 16$	Address Valid to ALE Falling Edge	9		ns
$t_{VP} = ¼\,t_C - 10$	Address Hold from ALE Falling Edge	15		ns
$t_{WAIT} = t_C = WS$	Wait State Period	100		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1052	kHz
t_{XIN}	Pulse Width for Timer Inputs	40		ns
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured paramenters, and may be used for system design with a very high confidence level.

Note: Measurement of AC specifications is done with external clock drive on CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF else AC measurements will be skewed.

CPU Read Cycle Timing f_C = 20 MHz with One Wait State (See *Figure 1*)

Symbol	Parameter	Min	Max	Units
$t_{ARR} = ½\,t_C - 20$	ALE Falling Edge to $\overline{RD}$ Falling Edge	30		ns
$t_{RW} = ¼\,t_C + WS - 10$	$\overline{RD}$ Pulse Width	115		ns
$t_{DR} = t_C - 15$	Data Hold after Rising Edge of $\overline{RD}$	0	85	ns
$t_{ACC} = t_C + WS - 55$	Address Valid to Input Data Valid		145	ns
$t_{RD} = ¼\,t_C + WS - 35$	$\overline{RD}$ Falling Edge to Input Data Valid		90	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	95		ns

Note: Minimum and Maximum values are calculated from maximum operating frequency.

CPU Write Cycle Timing f_C = 20 MHz with One Wait State (See *Figure 2*)

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 20$	ALE Falling Edge to $\overline{WR}$ Falling Edge	30		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{WR}$ Pulse Width	160		ns
$t_{HW} = \frac{1}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{WR}$	10		ns
$t_V = \frac{1}{2} t_C + WS - 40$	Data Valid before Rising Edge of $\overline{WR}$	110		ns

Note: Bus output (Port A) C_L = 100 pF, CK2 output C_L = 50 pF, other outputs C_L = 80 pF. AC Parameters are tested using DC Characteristics and non CMOS outputs.

DMA Read Cycle Timing f_C = 20 MHz (See *Figure 1*)

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{2} t_C - 20$	ALE Falling Edge to $\overline{RD}$ Falling Edge	30		ns
$t_{RW} = \frac{3}{2} t_C - 15$	$\overline{RD}$ Pulse Width	135		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold After Rising Edge of $\overline{RD}$	0	60	ns
$t_{ACC} = \frac{9}{4} t_C - 75$	Address Valid to Input Data Valid		150	ns
$t_{RD} = \frac{3}{2} t_C - 35$	$\overline{RD}$ Falling Edge to Input Data Valid		115	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	95		ns
$t_{VP} = \frac{1}{2} t_C - 10$	Address Hold from ALE Falling Edge	40		ns

Note: Minimum and Maximum values are calculated from moderate operating frequency.

DMA Write Cycle Timing f_C = 20 MHz (See *Figure 2*)

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 20$	ALE Trailing Edge to $\overline{WR}$ Falling Edge	30		ns
$t_{WW} = \frac{3}{2} t_C - 15$	$\overline{WR}$ Pulse Width	135		ns
$t_{HW} = \frac{1}{2} t_C - 15$	Data Hold After Trailing Edge of $\overline{RD}$	35		ns
$t_V = \frac{3}{2} t_C - 50$	Data Valid before Trailing Edge of $\overline{WR}$	100		ns
$t_{VP} = \frac{1}{2} t_C - 10$	Address Hold from ALE Falling Edge	40		ns

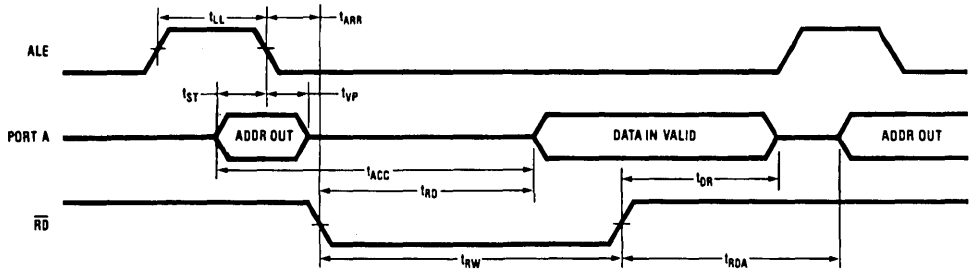
Note: Bus output (Port A) C_L = 100 pF, CK2 output C_L = 50 pF, other outputs C_L = 80 pF. AC Parameters are tested using DC Characteristics and non CMOS outputs.

Ready/Hold Timing f_C = 20 MHz with One Wait State

Symbol	Parameter	Min	Max	Units
$t_{DAR} = 1/4\ t_C + WS - 55$	Falling Edge of ALE to Falling Edge of $\overline{RDY}$		70	ns
$t_{RWP} = t_C$	$\overline{RDY}$ Pulse Width	100		ns
$t_{SALE} = 3/4\ t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	115		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	110		ns
$t_{HAD} = 7/4\ t_C + 50$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		225	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		200*	ns
t_{BF}	Bus Float before Falling Edge on $\overline{HLDA}$	0		ns
$t_{BE} = 3/4\ t_C + 50$	Bus Enable from Rising Edge of $\overline{HLDA}$		125	ns

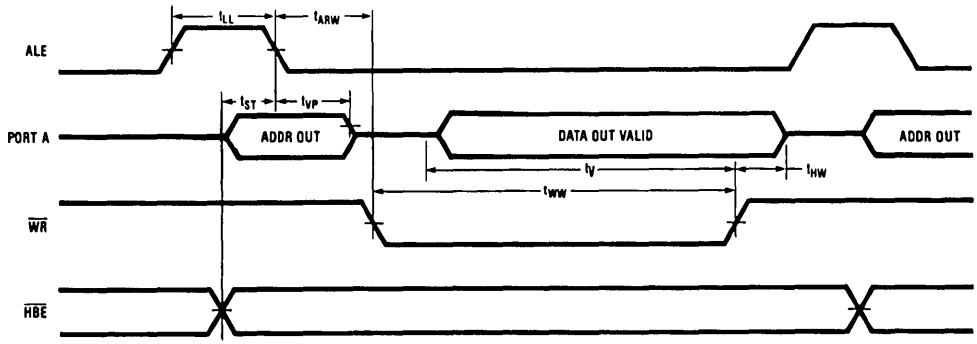
Note: t_{HAE} may be as long as $(3t_C + 4ws + 72t_C + 90)$ depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value tested is for the optimal case.

Timing Waveforms



TL/DD/8802-22

FIGURE 1. CPU and DMA Read Cycles

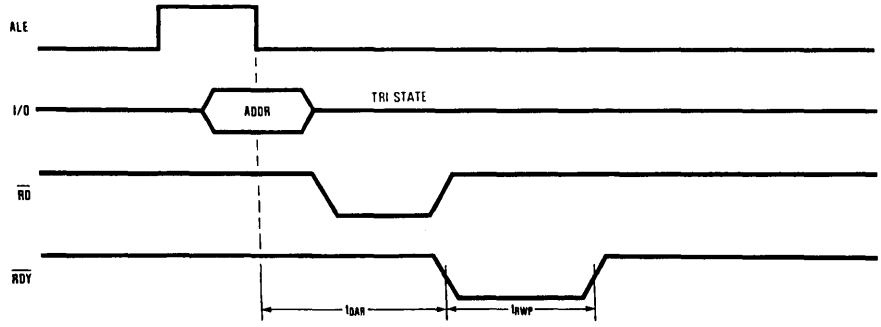


TL/DD/8802-21

FIGURE 2. CPU and DMA Write Cycles

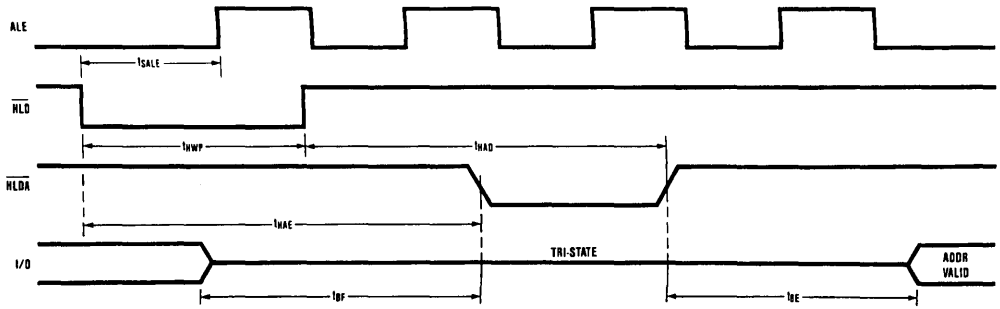
4

Timing Waveforms (Continued)



TL/DD/8802-4

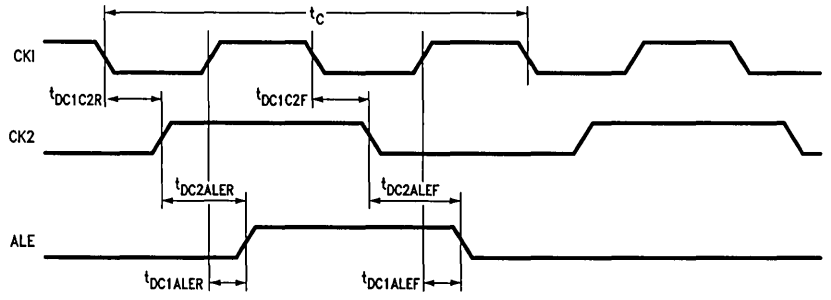
FIGURE 3. Ready Mode Timing



TL/DD/8802-5

FIGURE 4. Hold Mode Timing

Timing Waveforms (Continued)



TL/DD/8802–23

FIGURE 5. CKI, CK2 ALE Timing Diagram

Pin Descriptions

I/O PORTS

Port A is a 16-bit multiplexed address/data bus used for accessing external program and data memory. Four associated bus control signals are available on port B. The Address Latch Enable (ALE) signal is used to provide timing to demultiplex the bus. Reading from and writing to external memory are signalled by RD* and WR* respectively. External memory can be addressed as either bytes or words with the decoding controlled by two lines, Bus High Byte enable (HBE*) and Address/Data Line 0 (A0).

Port B is a 16-bit port, with 12 bits of bidirectional I/O similar in structure to port A. Pins B10, B11, B12 and B15 are the control bus signals for the address/data bus. Port B may also be configured via a function register BFUN to individually allow each bidirectional I/O pin to have an alternate function.

B0:	TDX	UART Data Output
B1:	CFLG1	Closing Flag Output for HDLC #1 Transmitter
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	HLDA*	Hold Acknowledge Output
B8:	TS0	Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	ALE	Address Latch Enable Output for Address/Data Bus
B11:	WR*	Address/Data Bus Write Output
B12:	HBE*	High Byte Enable Output for Address/Data Bus
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	RD*	Address/Data Bus Read Output

When operating in the extended memory addressing mode, four bits of port B can are used as follows—

B8:	BS0	Memory bank switch output 0 (LSB)
B9:	BS1	Memory bank switch output 1
B13:	BS2	Memory bank switch output 2
B14:	BS3	Memory bank switch output 3 (MSB)

Port I is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

I0:	HCK2	HLDC #2 Clock Input
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture
I3:	INT3	Maskable Interrupt/Input Capture
I4:	INT4/RDY	Maskable Interrupt/Input Capture/ Ready Input
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:	HCK1	HDLC #1 Clock/Serial Decoder Clock Input

Port D is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

D0:	REN1/FS/ RHCK1	Receiver #1 Enable/Serial Decoder Frame Sync Input/Receiver #1 Clock Input
D1:	TEN1	Transmitter #1 Enable Input
D2:	REN2/ RHCK2	Receiver #2 Enable Input/Receiver #2 Clock Input
D3:	TEN2	Transmitter #2 Enable Input
D4:	RX1	Receiver #1 Data Input
D5:	TX1	Transmitter #1 Data Output
D6:	RX2	Receiver #2 Data Input
D7:	TX2	Transmitter #2 Data Output

Note: Any of these pins can be read by software. Therefore, unused functions can be used as general purpose inputs, notably external enable lines when the internal serial decoder is used (see SERIAL DECODER/ENABLE CONFIGURATION REGISTER).

Port R is an 8-bit bidirectional I/O port available for general purpose I/O operations. Port R has a direction register to enable each separate pin to be individually defined as an input or output. It has a data register which contains the value to be output. In addition, the Port R pins can be read directly using the Port R pins address.

Pin Descriptions (Continued)

POWER SUPPLIES

V_{CC} Positive Power Supply (two pins)

GND Ground for On-Chip Logic

DGND Ground for Output Buffers

CLOCK PINS

CKI The System Clock Input

CKO The System Clock Output (Inversion of CKI)

Pins CKI and CKO are usually connected across an external crystal.

CK2 Clock Output (CKI divided by 2)

OTHER PINS

WO This is an active low open drain output which signals an illegal situation has been detected by the Watch Dog logic.

ST1 Bus Cycle Status Output indicates first opcode fetch.

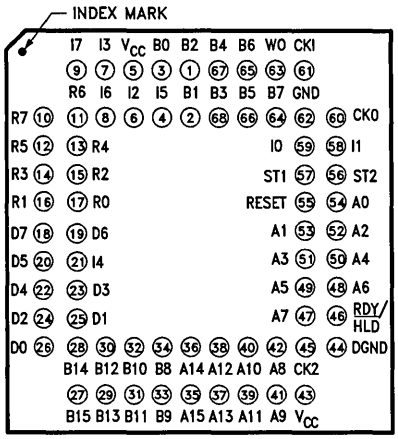
ST2 Bus Cycle Status Output indicates machine states (skip and interrupt).

RESET Active low input that forces the chip to restart and sets the ports in a TRI-STATE mode.

RDY/HLD Has two uses, selected by a software bit. This pin is either a READY input to extend the bus cycle for slower memories or a HOLD-REQUEST input to put the bus in a high impedance state for external DMA purposes. In the second case the I4 pin becomes the READY input.

Connection Diagram

Pin Grid Array

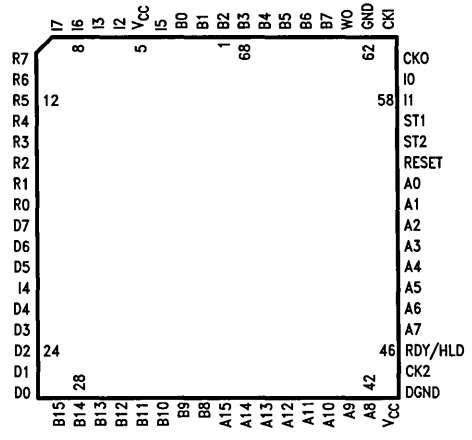


TL/DD/8802-24

Top View

Order Number HPC16400EL or HPC16400U
See NS Package Number EL68A or U68A

Plastic and Leadless Chip Carriers



TL/DD/8802-17

Top View

Order Number HPC16400E or V
See NS Package Number E68B or V68A

Wait States

The HPC16400 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals. The DMA always uses one Wait State, independent of the value selected in the PSW.

Power Save Modes

Two power saving modes are available on the HPC16400: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16400 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16400 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the RESET or the NMI. The RESET input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16400 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. The HPC16400 resumes normal operation upon timer T0 overflow. As with the HALT mode, the processor is returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization.

HPC16400 Interrupts

Complex interrupt handling is easily accomplished by the HPC16400's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table I.

TABLE I. Interrupts

Vector/ Address	Interrupt Source	Arbitration Ranking
FFFF|FFFE	Reset	0
FFFD|FFFC	Nonmaskable Ext (NMI)	1
FFFB|FFFA	External on I2	2
FFF9|FFF8	External on I3	3
FFF7|FFF6	I4 + HDLC/DMA Error	4
FFF5|FFF4	Internal on Timers	5
FFF3|FFF2	Internal on UART	6
FFF1|FFF0	End of Message (EOM)	7

The 16400 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. Interrupts are serviced after the current instruction is completed except for the RESET which is serviced immediately.

The NMI interrupt will immediately stop DMA activity-byte transfers in progress will finish thereby allowing an orderly transition to the interrupt service vector (see DMA description). The HDLC channels continue to operate, and the user must service data errors that might have occurred during the NMI service routine.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately.

RESET is a level-sensitive interrupt. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 can be software selected to be rising or falling edge.

Interrupt Control Registers

The HPC16400 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to be serviced, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16400 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable (GIE) bit is reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack, set the GIE bit and return to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 6* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A, B (except B12), D and R in the TRI-STATE condition. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location.

Timer Overview

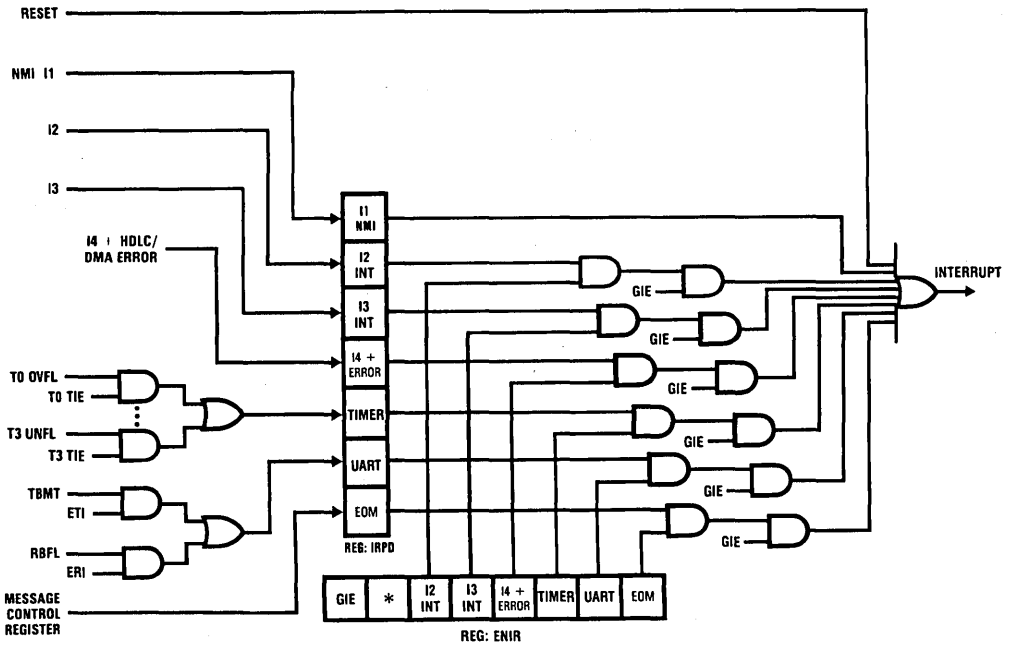
The HPC16400 contains a powerful set of flexible timers enabling the HPC16400 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16400 contains four 16-bit timers. Three of the timers have an associated 16-bit register. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watch Dog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register T0CON configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 7*).

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 8*).

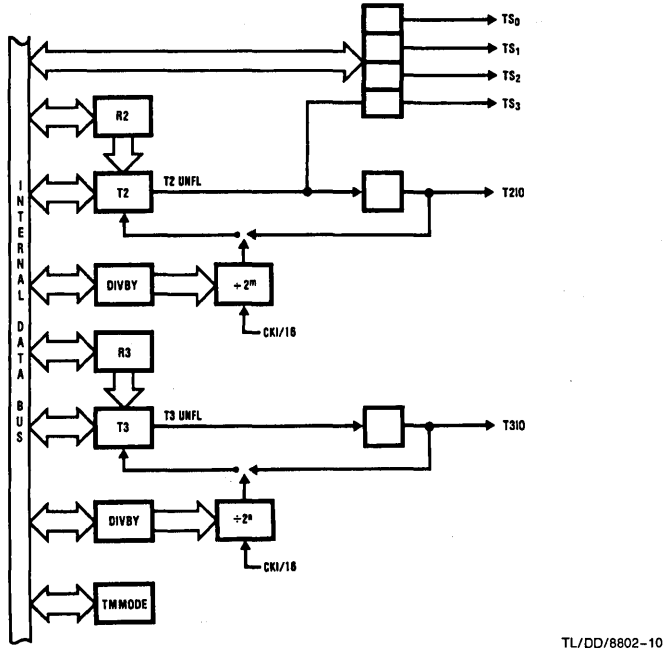
The timers T1 through T3 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

Timer Overview (Continued)



TL/DD/8802-8

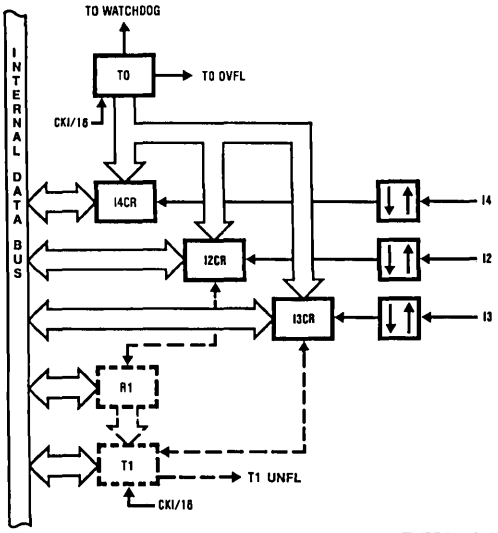
FIGURE 6. Interrupt Enable Logic



TL/DD/8802-10

FIGURE 8. Timers T2–T3 Block

Timer Overview (Continued)



TL/DD/8802-9

FIGURE 7. Timers T0–T1 Block

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16400 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 8*). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to ½ the frequency of the source used for clocking the timer.

Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch and enable interrupts from timers T0 through T3.

Timer Applications

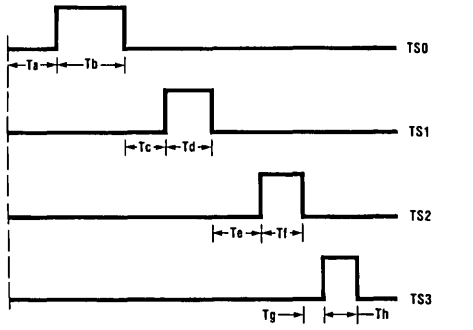
The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16400.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.

TL/DD/8802-12

FIGURE 9. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 10* is an example of synchronous pulse train generation.



TL/DD/8802-13

FIGURE 10. Synchronous Pulse Generation

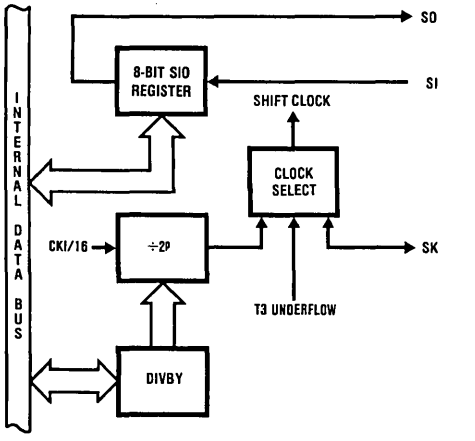
Watch Dog Logic

The Watch Dog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watch Dog logic are potentially infinite loops. Should the Watch Dog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. The illegal condition forces the Watch Out (WO) pin low. The WO pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 11*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., ISDN Transceivers, A/D converters, display drivers, EEPROMs).



TL/DD/8802-14

FIGURE 11. MICROWIRE/PLUS

MICROWIRE/PLUS Operation

The HPC16400 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16400 is the master or slave. The shift clock is generated when the HPC16400 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16400 is configured as a slave. When the HPC16400 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz with CKI at 16 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is shifted out on the falling edge of the SK clock. Serial data on the SI pin is latched in on the rising edge of the SK clock.

HPC16400 UART

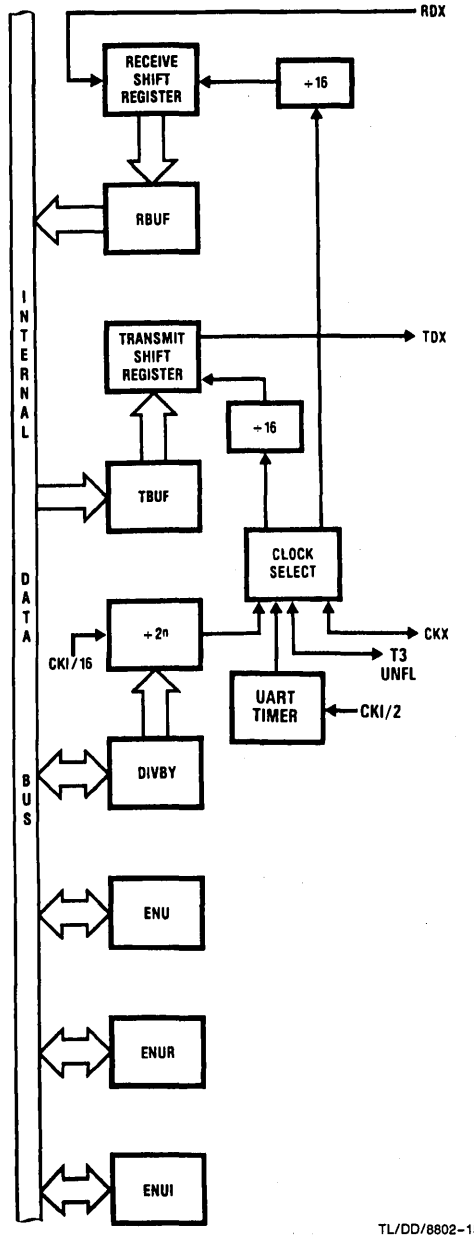
The HPC16400 contains a software programmable UART. The UART (see *Figure 12*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register, a special dedicated timer. The baud rate may be selected from a range of 8 baud to 208.3 kbaud. Without having to select a special baud rate crystal, all standard baud rates from 75 baud to 38.4 kbaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC16400 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16400 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16400 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.



TL/DD/8802-15

FIGURE 12. UART Block Diagram

UART Wake-up Mode (Continued)

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16400 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

Programmable Serial Decoder Interface

The programmable serial decoder interface allows the two HDLC channels to be used with devices employing several popular serial protocols for point-to-point and multipoint data exchanges. These protocols combine the 'B' and 'D' channels onto common pins—received data, transmit data, clock and Sync, which normally occurs at an 8 KHz rate and provides framing for the particular protocol.

The decoder uses the serial link clock and Sync signals to generate internal enables for the 'D' and 'B' channels, thereby allowing the HDLC channels to access the appropriate channel data from the multiplexed link.

HDLC Channel Description

HDLC/DMA Structure

HDLC 1		HDLC 2	
HDLC1 Receive	HDLC1 Transmit	HDLC2 Receive	HDLC2 Transmit
DMAR1	DMAT1	DMAR2	DMAT2

GENERAL INFORMATION

Both HDLC channels on the HPC16400 are identical and operate up to 4.65 Mbps. When used in an ISDN basic access application, HDLC channel #1 has been designated for use with the 16 Kbps D-channel or either B channel and HDLC #2 can be used with either of the 64 Kbps B-channels. If the 'D' and 'B' channels are present on a common serial link, the programmable serial decoder interface generates the necessary enable signals needed to access the D and B channel data.

LAPD, the Link Access Protocol for the D channel is derived from the X.25 packet switching LAPB protocol. LAPD specifies the procedure for a terminal to use the D channel for the transfer of call control or user-data information. The procedure is used in both point-to-point and point-to-multipoint configurations. On the 16400, the HDLC controller contains user programmable features that allow for the efficient processing of LAPD Information.

HDLC Channel Pin Description

Each HDLC channel has the following pins associated with it.

HCK — HDLC Channel Clock Input Signal.

RX — Receive Serial Data Input. Data latched on the negative HCK edge.

REN/RHCK — HDLC Channel Receiver Enable Input/Receiver Clock Input.

TEN — HDLC Channel Transmitter Enable Input.

TX — Transmit Serial Data Output. Data clocked out on the positive HCK edge. Data (not including CRC) is sent LSB first. TRI-STATE when transmitter not enabled.

HDLC Functional Description

TRANSMITTER DESCRIPTION

Data information is transferred from external memory through the DMA controller into the transmit buffer register from where it is loaded into a 8-bit serial shift register. The CRC is computed and appended to the frame prior to the closing flag being transmitted. Data is output at the TX output pin. If no further transmit commands are given the transmitter sends out continous flags, aborts, or the idle pattern as selected by the control register.

An interrupt is generated when the transmit shift register is empty or on a transmit error condition. An assoicated transmit status register will contain the status information indicating the specific interrupt source.

TRANSMITTER FEATURES

Interframe fill: the transmitter can send either continuous '1's or repeated flags or aborts between the closing flag of one packet and the opening flag of the next. When the CPU commands the transmitter to open a new frame, the interframe fill is terminated immediately.

Abort: the 7 '1's abort sequence will be immediately sent on command from the CPU or on an underrun condition in the DMA. If required it may be followed by a new opening flag to send another packet.

Bit/Byte boundaries: The message length between packet headers may have any number of bits and is not confined to an integral number of bytes. Three bits in the control register are used to indicate the number of valid bits in the last byte. These bits are loaded by the users software.

RECEIVER DESCRIPTION

Data is input to the receiver on the RX pin. The receive clock can be externally input at either the HCK pin or the REN/RHCK pin.

Incoming data is routed through one of several paths depending on whether it is the flag, data, or CRC.

Once the receiver is enabled it waits for the opening flag of the incoming frame, then starts the zero bit deletion, addressing handling and CRC checking. All data between the flags is shifted through two 8-bit serial shift registers before being loaded into the buffer register. The user programmable address register values are compared to the incoming data while it resides in the shift registers. If an address match occurs or if operating in the transparent address recognition mode, the DMA channel is signaled that attention is required and the byte is transferred by it to external memory. Appropriate interrupts are generated to the CPU on the reception of a complete frame, or on the occurance of a frame error.

There are two sources for the receive channel enable signal. It can be internally generated from the serial decoder interface or it can be externally enabled.

The receive interrupt, in conjunction with status data in the control registers allows interrupts to be generated on the following conditions—CRC error, receive error and receive complete.

RECEIVER FEATURES

Flag sharing: the closing flag of one packet may be shared as the opening flag of the next. Receiver will also be able to share a zero between flags—011111101111110 is a valid two flag sequence for receive (not transmit).

HDLC Functional Description (Continued)

Interframe fill: the receiver automatically accepts either repeated flags, repeated aborts, or all '1's as the interframe fill.

Idle: Reception of successive flags as the interframe fill sequence to be signaled to the user by setting the Flag bit in the Receive status register.

Short Frame Rejection: Reception of greater than 2 bytes but less than 4 bytes between flags will generate a frame error, terminating reception of the current frame and setting the Frame Error (FER) status bit in the Receive Control and Status register. Reception of less than 2 bytes will be ignored.

Abort: the 7 '1's abort sequence (received with no zero insertion) will be immediately recognized and will cause the receiver to reinitialize and return to searching the incoming data for an opening flag. Reception of the abort will cause the abort status bit in the Interrupt Error Status register to be set.

Bit/Byte boundaries: The message length between packet headers may have any number of bits and it is not confined to an integral number of bytes. Three bits in the status register are used to indicate the number of valid bits in the last byte.

Addressing: Two user programmable bytes are available to allow frame address recognition on the two bytes immediately following the opening flag. When the received address matches the programmed value(s), the frame is passed through to the DMA channel. If no match occurs, the received frame address information is disregarded and the receiver returns to searching for the next opening flag and the address recognition process starts anew.

Support is provided to allow recognition of the broadcast address sequence of seven consecutive 1's. Additionally, a transparent mode of operation is available where no address decoding is done.

HDLC INTERRUPT CONDITIONS

The end of message interrupt (EOM) indicates that a complete frame has been received or transmitted by the HDLC controller. Thus, there are four separate sources for this interrupt, two each from each HDLC channel. The Message Control Register contains the pending bits for each source.

The HDLC/DMA error interrupt groups several related error conditions. Error conditions from both transmit/receiver channels can cause this interrupt, and the possible sources each have a status bit in the error status register that is set on the occurrence of an error. The bit must then be serviced by the user.

HDLC CHANNEL CLOCK

Each HDLC channel uses the falling edge of the clock to sample the receive data. Outgoing transmit data is shifted out on the rising edge of the external clock. The maximum data rate when using the externally provided clocks is 4.65 Mb/s.

CYCLIC REDUNDACY CHECK

There are two standard CRC codes used in generating the 16-bit Frame Check Sequence (FCS) that is appended to the end of the data frame. Both codes are supported and the user selects the error checking code to be used through software control (HDLC control reg). The two error checking polynomials available are:

(1) CRC—16 (x16 + x15 + x2 + 1)

(2) CCITT CRC (x16 + x12 + x5 + 1)

SYNCHRONOUS BYPASS MODE

When the BYPAS bit is set in the HDLC control register, all HDLC framing/formatting functions for the specified HDLC channel are disabled.

This allows byte-oriented data to be transmitted and received synchronously thus "bypassing" the HDLC functions.

LOOP BACK OPERATIONAL MODE

The user has the ability, by setting the appropriate bit in the register to internally route the transmitter output to the receiver input, and to internally route the RX pin to the TX pin.

DMA Controller*

GENERAL INFORMATION

The HPC16400 uses Direct Memory Access (DMA) logic to facilitate data transfer between the 2 full Duplex HDLC channels and external packet RAM. There are four DMA channels to support the four individual HDLC channels. Control of the DMA channels is accomplished through registers which are configured by the CPU. These control registers define specific operation of each channel and changes are immediately reflected in DMA operation. In addition to individual control registers, a global control bit (MSS in Message Control Register) is available so that the HDLC channels may be globally controlled.

The DMA issues a bus request to the CPU when one or more of the individual HDLC channels request service. Upon receiving a bus acknowledge from the CPU, the DMA completes all requests pending and any requests that may have occurred during DMA operation before returning control to the CPU. If no further DMA transfers are pending, the DMA relinquishes the bus and the CPU can again initiate a bus cycle.

Four memory expansion bits have been added for each of the four channels to support data transfers into the expanded memory bank areas.

The DMA has priority logic for a DMA requesting service. The priorities are:

1st priority .....Receiver channel 1
2nd priority .....Transmit channel 1
3rd priority .....Receive channel 2
4th priority .....Transmit channel 2

RECEIVER DMA OPERATION

The receiver DMA consists of a shift register and two buffers. A receiver DMA operation is initiated by the buffer registers. Once a byte has been placed in a buffer register from the HDLC, it generates a request and upon obtaining control of the bus, the DMA places the byte in external memory.

RECEIVER REGISTERS

All the following registers are Read/Write

A. Frame Length Register

This user programmable 16-bit register contains the maximum number of bytes to be placed in a data "block". If this number is exceeded, a Frame Too Long (FTLR1, FTLR2) error is generated. This register is decremented by one each Receiver DMA cycle.

*The specific registers and/or register names may have changed. Please contact the factory for updated information.

DMA Controller (Continued)

B. CNTRL ADDR 1
DATA ADDR 1
CNTRL ADDR 2
DATA ADDR 2

For split frame operation, the CNTRL ADDR register contains the external memory address where the Frame Header (Control & Address fields) are to be stored and the DATA ADDR register contains an equivalent address for the Information field.

For non-split frame operation, the CNTRL and DATA ADDR registers each contain the external memory address for the entire frame.

TRANSMITTER DMA OPERATION

The transmitter DMA consists of a shift register and two buffers. A transmitter DMA cycle is initiated by the TX data buffers. The TX data buffers generate a request when either one is empty and the DMA responds by placing a byte in the buffer. The HDLC transmitter can then accept the byte to send when needed, upon which the DMA will issue another request, resulting in a subsequent DMA cycle.

TRANSMITTER REGISTERS

The following registers are Read/Write:

A. Field Address 1 (FA1)
Bytes Field 1 (NBF1)
Field Address (FA2)
Bytes Field 2 (NBF2)

FA1 and FA2 are starting addresses of blocks of information to transmitter.

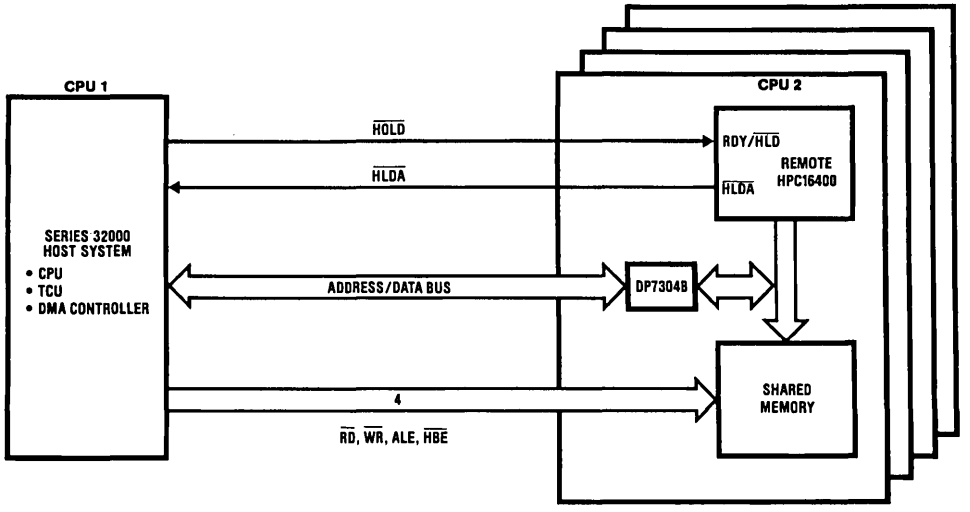
NBF1 and NBF2 are the number of bytes in the block to be transmitted.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16400 supports shared memory access with two pins. The pins are the RDY/$\overline{HLD}$ input pin and the $\overline{HLDA}$ output pin. The user can software select either the Hold or Ready function on the RDY/HLD pin by the state of a control bit. The $\overline{HLDA}$ output is multiplexed onto port B.

The host uses DMA to interface with the HPC16400. The host initiates a data transfer by activating the $\overline{HLD}$ input of the HPC16400. In response, the HPC16400 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{HLDA}$) from the HPC16400 indicating that the sytem bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16400 resumes normal operations.

Figure 13 illustrates an application of the shared memory interface between the HPC16400 and a Series 32000 system.



TL/DD/8802-16

FIGURE 13. Shared Memory Application: HPC16400 Interface to Series 32000 System

Memory

The HPC16400 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 256 bytes of RAM available on the chip itself.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16400 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC16400 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table II.

Extended Memory Addressing

If more than 64k of addressing is desired in a HPC16400 system, on board bank select circuitry is available that allows four I/O lines of Port B (B8, B9, B13, B14) to be used in extending the address range. This gives the user a main routine area of 32k and 16 banks of 32k each for subroutine and data, thus getting a total of 544k of memory.

Note: If all four lines are not needed for memory expansion, the unused lines can be used as general purpose inputs.

The Extended Memory Addressing mode is entered by setting the EMA control bit in the Message Control Register. If this bit is not set, the port B lines (B8, B9, B13, B14) are available as general purpose I/O or synchronous outputs as selected by the BFUN register.

The main memory area contains the interrupt vectors & service routines, stack memory, and common memory for the bank subroutines to use. The 16 banks of memory can contain program or data memory (note: since the on chip resources are mapped into addresses 0000-01FF, the first 512 bytes of each bank are not usable).

TABLE II. Memory Map

FFFF–FFF0 FFEF–FFD0	Interrupt Vectors JSRP Vectors	
FFCF–FFCE : : 0201–0200	External Expansion	USER MEMORY
01FF–01FE : : 01C1–01C0	On Chip RAM	USER RAM
01B8 01B6 01B4 01B2 01B0	Error Status Receiver Status HDLC Cntrl Recr Addr Comp Reg 2 Recr Addr Comp Reg 1	HDLC # 2
01A8 01A6 01A4 01A2 01A0	Error Status Receiver Status HDLC Cntrl Recr Addr Comp Reg 2 Recr Addr Comp Reg 1	HDLC # 1
0195–0194	Watch Dog Register	Watch Dog Logic
0193–0192 0191–0190 018F–018E 018D–018C 018B–018A 0189–0188 0187–0186 0185–0184 0183–0182 0181–0180	T0CON Register TMMODE Register DIVBY Register T3 Timer R3 Register T2 Timer R2 Register I2CR Register/ R1 I3CR Register/ T1 I4CR Register	Timer Block T0–T3
017F–017E 017D–017C	UART Counter UART Register	
0179–0178 0177–0176 0175–0174 0173–0172 0171–0170	# Bytes 2 Field Addr 2 # Bytes 1 Field Addr 1 Xmit Cntrl & Status	DMAT # 2 (Xmit)
016B–016A 0169–0168 0167–0166 0165–0164 0163–0162 0161–0160	Frame Length Data Addr 2 Cntrl Addr 2 Data Addr 1 Cntrl Addr 1 Recv Cntrl & Status	DMAR # 2 (Recv)
0159–0158 0157–0156 0155–0154 0153–0152 0151–0150	# Bytes 2 Field Addr 2 # Bytes 1 Field Addr 1 Xmit Cntrl & Status	DMAT # 1 (Xmit)
014B–014A 0149–0148 0147–0146 0145–0144 0143–0142 0141–0140	Frame Length Data Addr 2 Cntrl Addr 2 Data Addr 1 Cntrl Addr 1 Recv Cntrl & Status	DMAR # 1 (Recv)
0128 0126 0124 0122 0120	ENUR Register TBUF Register RBUF Register ENUI Register ENU Register	UART
010E 010C 010A 0106 0104 0102 0100	Port R Pins DIR R Register Port R Data Register Serial Decoder/Enable Configuration Reg Message Pending Message Control Port D Pins	PORTS R & D
00F5–00F4 00F3–00F2 00E3–00E2	BFUN Register DIR B Register Port B	PORT B
00DE 00DD–00DC 00D8 00D6 00D4 00D2 00D0	Microcode ROM Dump Halt Enable Register Port I Input Register SIO Register IRCD Register IRPD Register ENIR Register	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00CF–00CE 00CD–00CC 00CB–00CA 00C9–00C8 00C7–00C6 00C5–00C4 00C3–00C2 00C1–00C0	X Register B Register K Register A Register PC Register SP Register (Reserved) PSW Register	HPC16040 CORE REGISTERS
00BF–00BE : : 0001–0000	On Chip RAM	USER RAM

HPC16400 CPU

The HPC16400 CPU has a 16-bit ALU and six 16-bit registers.

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the stack pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16400 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action	
ARITHMETIC INSTRUCTIONS			
ADD	Add	MA + Meml → MA	carry → C
ADDS	Add short imm8	MA + imm8 → MA	carry → C
ADC	Add with carry	MA + Meml + C → MA	carry → C
DADC	Decimal add with carry	MA + Meml + C → MA (Decimal)	carry → C
SUBC	Subtract with carry	MA − Meml + C → MA	carry → C
DSUBC	Decimal subtract w/carry	MA − Meml + C → MA (Decimal)	carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C	
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C	
DIVD	Divide Double Word (unsigned)	(x8 MA)/Meml → MA, rem → X, 0 → K	carry → C
IFEQ	If equal	Compare MA & Meml, Do next if equal	
IFGT	If greater than	Compare MA & Meml, Do next if MA → Meml	
AND	Logical and	MA and Meml → MA	
OR	Logical or	MA or Meml → MA	
XOR	Logical exclusive-or	MA xor Meml → MA	
MEMORY MODIFY INSTRUCTIONS			
INC	Increment	Mem + 1 → Mem	
DECSZ	Decrement, skip if 0	Mem − 1 → Mem, Skip next if Mem = 0	
BIT INSTRUCTIONS			
SBIT	Set bit	1 → Mem.bit (bit is 0 to 7 immediate)	
RBIT	Reset bit	0 → Mem.bit	
IFBIT	If bit	If Mem.bit is true, do next instr.	
MEMORY TRANSFER INSTRUCTIONS			
LD	Load	Meml → MA	
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X	
ST	Store to Memory	MA → Mem	
X	Exchange	A ↔ Mem; Mem ↔ Mem	
	Exchange, incr/decr X	A ↔ Mem(X), X ±1 (or 2) → X	
PUSH	Push Memory to Stack	W → W(SP), SP + 2 → SP	
POP	Pop Stack to Memory	SP − 2 → SP, W(SP) → W	
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K	
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A,B ±1 (or 2) → B, Skip next if B greater/less than K	
REGISTER LOAD IMMEDIATE INSTRUCTIONS			
LD A	Load A immediate	imm → A	
LD B	Load B immediate	imm → B	
LD K	Load K immediate	imm → K	
LD X	Load X immediate	imm → X	
LD BK	Load B and K immediate	imm → B,imm → K	
ACCUMULATOR AND C INSTRUCTIONS			
CLR A	Clear A	0 → A	
INC A	Increment A	A + 1 → A	
DEC A	Decrement A	A − 1 → A	
COMP A	Complement A	1's complement of A → A	
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0	
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C	
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C	
SHR A	Shift A right	0 → A15 → ... → A0 → C	
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0	
SC	Set C	1 → C	
RC	Reset C	0 → C	
IFC	IF C	Do next if C = 1	
IFNC	IF not C	Do next if C = 0	

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP),SP+2 → SP W(table #) → PC
JSR	Jump subroutine relative	PC → W(SP),SP+2 → SP,PC+ # → PC (#is +1024 to −1023)
JSRL	Jump subroutine long	PC → W(SP),SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +256 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC ← PC + 1
RET	Return	SP−2 → SP,W(SP) → PC
RETS	Return then skip next	SP−2 → SP,W(SP) → PC, & skip
RETI	Return from interrupt	SP−2 → SP,W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8 or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
Meml is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir. (B)	Reg Indir. (X)	Direct	Indir	Index	Immed.	Direct *	Direct **	Immed. *	Immed. **
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Memory Usage (Continued)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC	C	1
RC	C	1
IFC	C	1
IFNC	C	1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETS	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16400 has been designed to be extremely code-efficient. The HPC16400 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16400, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16400 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16400 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16400 supplies 8-bit byte capability for 2-digit variables and literal variables.

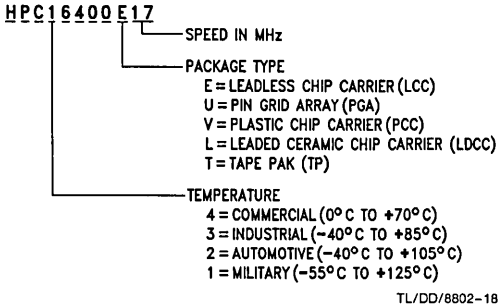
MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16400 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16400 has been generally used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.



TL/DD/8802-18

FIGURE 15. HPC Family Part Numbering Scheme

EXAMPLES

HPC46400V17—Commercial temp (0° to +70°C), PCC

HPC16400E17—Military temp (−55°C to +125°C), LCC

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcontroller On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self-contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host systems using PC-DOS communicating via a RS-232 port.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBMR	Relocatible Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001
	424410897-001	Users Manual		

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the MOLE (Microcontroller On Line Emulator) applications group. It consists of both an electronic bulletin board information system and a method by which applications can take control of a MOLE Development System at a remote site via modem in order to resolve any problems.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The user needs as a minimum, a Dumb terminal, 300 or 1200 baud Modem, and a telephone.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains:
- Dial-A-Helper Users Manual
- Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

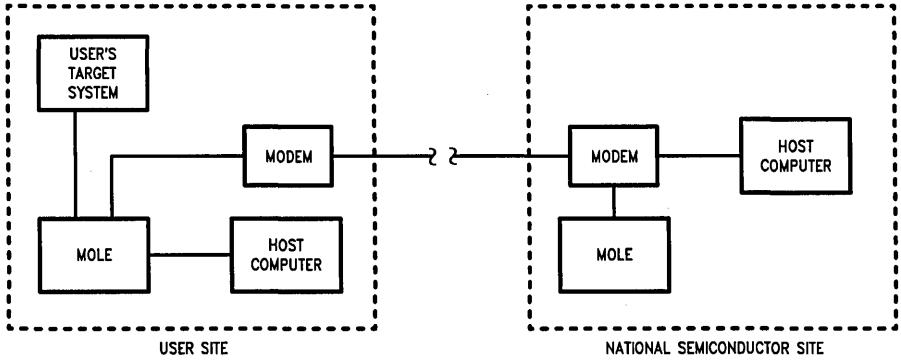
Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in getting a MOLE to operate in a particular mode or something peculiar is occurring, he can contact us via his system and modem. He can leave messages on our electronic bulletin board, which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting customer's system to respond. Both parties see exactly what is occurring, as it is happening.

This allows us to respond in minutes when applications help is needed.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Set-Up: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs, 7 Days

DIAL-A-HELPER



TL/DD/8802–20

National Semiconductor

PRELIMINARY

HPC16900/HPC26900/HPC36900/HPC46900 PEARL Port Expander And Re-creation Logic

General Description

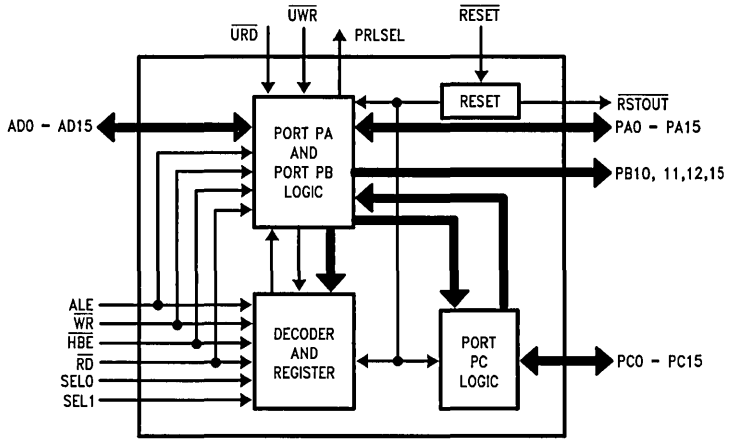
The PEARL is a peripheral device which re-creates Port A and four bits of Port B when used with HPC family microcontrollers. An additional 16-bit port (Port PC) is configured as either a 16-bit latched address bus or as 16 general purpose I/O pins. The PEARL is intended for port expansion when the user requires Port A on the HPC to serve as an address/data bus.

The PEARL can also serve as the interface to a host controller when the HPC is used as a Universal Peripheral Interface with its own dedicated address/data bus.

Features

- Provides up to 36 I/O pins
- Multiplexed address/data bus
- Interfaces to other microprocessor families
- Supports UPI (Universal Peripheral Interface) mode
- Fabricated in 2 micron, double-metal CMOS
- 3.0V to 5.5V operation
- Commercial (0°C to +70°C)
  - Industrial (−40°C to +85°C)
  - Automotive (−40°C to +105°C)
  - Military (−55°C to +125°C) temperature ranges
- 68 pin package
- Supports 17 MHz HPC operation
- As many as four PEARL chips may be used in parallel without additional interface chips.

Block Diagram



TL/DD/9122–1

4

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	V_{CC} + 0.5V to GND − 0.5V

ESD rating is to be determined.

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absoulte maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46900

Symbol	Parameter	Conditions	Min	Max	Units
I_{CC}	Supply Current	V_{CC} = 5.5V, t_{CY} = 130 ns		20	mA
	Static Current	V_{CC} = 5.5V (Note 1)		100	μA
INPUT VOLTAGE LEVELS ($\overline{RESET}$)					
V_{IH1}	Logic High		0.9 V_{CC}		V
V_{IL1}	Logic Low			0.1 V_{CC}	V
INPUT VOLTAGE LEVELS (ALL OTHER INPUTS)					
V_{IH2}	Logic High		0.7 V_{CC}		V
V_{IL2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current: All Other Inputs			±1	μA
I_{LI}	Input Leakage Current: PB12			±25	μA
OUTPUT VOLTAGE LEVELS (CMOS OPERATION)					
V_{OH1}	Logic High	I_{OH} = −10 μA	V_{CC} − 0.1		V
V_{OL1}	Logic Low	I_{OL} = 10 μA		0.1	V
V_{OH2}	Output Drive PC0–PC15, PRLSEL	I_{OH} = −4 mA	2.4		V
V_{OL2}		I_{OL} = 2 mA		0.4	V
V_{OH3}	Output Drive All Other Outputs	I_{OH} = −7 mA	2.4		V
V_{OL3}		I_{OL} = 3 mA		0.4	V

Note 1: $\overline{RESET}$ = GND; AD0–AD15 = V_{CC}; PA0–PA15 = V_{CC}; PC0–PC15 = V_{CC}; PB10,11,12,15 = V_{CC}, SEL0, SEL1 = GND; $\overline{ALE}$, $\overline{RD}$, $\overline{WR}$, $\overline{HBE}$ = V_{CC}.

AC Electrical Characteristics

V_{CC} = 5V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46900

Symbol	Parameter	Min	Max	Units
t_{ADPC}	AD0–AD12 to PC0–PC12 Latched Output		40	ns
t_{RSTF}	$\overline{RESET}$ Falling Edge to $\overline{NRSTOUT}$ Falling Edge		26	ns
t_{RSTR}	$\overline{RESET}$ Rising Edge to $\overline{NRSTOUT}$ Rising Edge		26	ns
t_{APS}	PRLSEL Delay after Address Valid at AD0–AD15		40	ns
t_{LL}	ALE Pulse Width	25		ns
t_{ST}	Address Valid to ALE Falling Edge	22		ns
t_{CY}	Rising Edge ALE to Rising Edge ALE Cycle	130		ns
t_{ALEPC}	AD13–AD15 to PC13–PC15 Latched Output		35	ns

AD and PA outputs C_L = 100 pF, PC outputs C_L = 50 pF, other outputs C_L = 80 pF.

AC Electrical Characteristics (Continued)

V_{CC} = 5V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46900

PEARL Register Read Timing

Symbol	Parameter	Min	Max	Units
t_{VPR}	Address Valid from ALE Trailing Edge Prior to $\overline{RD}$	13		ns
t_{AR}	ALE Falling Edge to $\overline{RD}$ Valid	13		ns
t_{RD}	$\overline{RD}$ Valid to Data Out Valid		40	ns
t_{RW}	$\overline{RD}$ Pulse Width	60		ns
t_{RDI}	Rising Edge of $\overline{RD}$ to Data Invalid	10	24	ns
t_{RA}	Rising Edge of $\overline{RD}$ to Rising Edge of ALE	10		ns

PEARL Register Write Timing

Symbol	Parameter	Min	Max	Units
t_{VPW}	Address Valid from ALE Falling Edge Prior to $\overline{WR}$	13		ns
t_{AW}	ALE Falling Edge to $\overline{WR}$ Valid	13		ns
t_V	Data Valid before Rising Edge of $\overline{WR}$	12		ns
t_{HW}	Data Hold after Rising Edge of $\overline{WR}$	11		ns
t_{WW}	$\overline{WR}$ Pulse Width	25		ns
t_{WA}	Rising Edge of $\overline{WR}$ to Rising Edge of ALE	11		ns

PEARL Port Output Timing

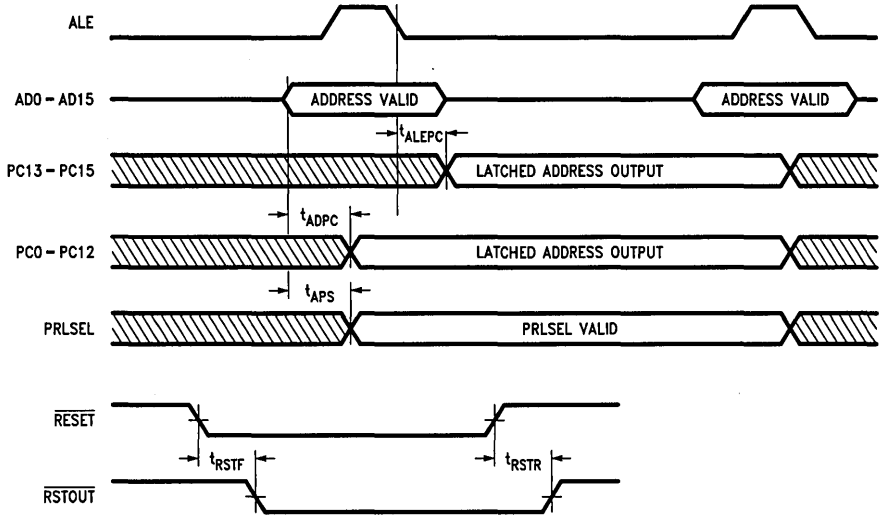
Symbol	Parameter	Min	Max	Units
t_{WPA}	$\overline{WR}$ Rising Edge to Port PA Data Valid		41	ns
t_{WPB}	$\overline{WR}$ Rising Edge to Port PB Data Valid		38	ns
t_{WPC}	$\overline{WR}$ Rising Edge to Port PC Data Valid		41	ns

UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{URD}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{URD}$	10		ns
t_{RPW}	$\overline{URD}$ Pulse Width	100		ns
t_{OE}	$\overline{URD}$ Falling Edge to Data Out Valid	60		ns
t_{OD}	Data Out Valid after Rising Edge of $\overline{URD}$	10	26	ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Rising Edge of $\overline{URD}$		40	ns
t_{WDW}	$\overline{UWR}$ Pulse Width	40		ns
t_{UDS}	Data In Valid before Rising Edge of $\overline{UWR}$	15		ns
t_{UDH}	Data In Valid after Rising Edge of $\overline{UWR}$	20		ns
t_A	$\overline{WRRDY}$ Delay from Rising Edge of $\overline{UWR}$		40	ns

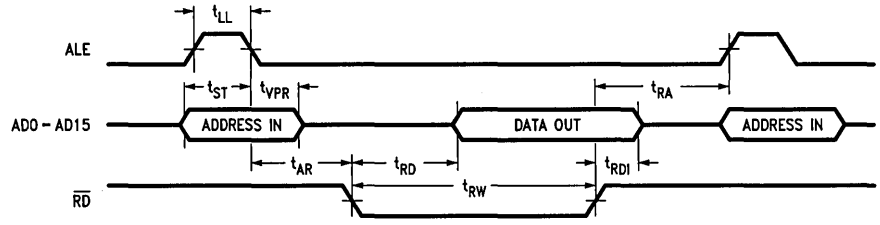
AD and PA outputs C_L = 100 pF, PC outputs C_L = 50 pF, other outputs C_L = 80 pF.

Timing Waveforms



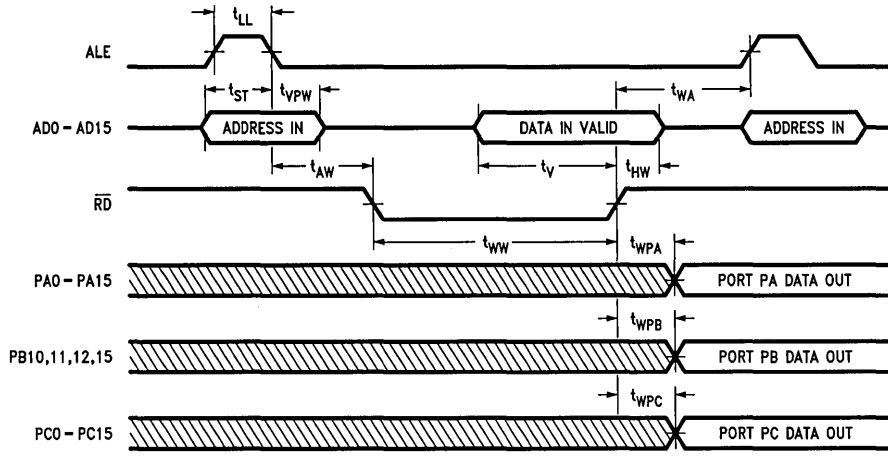
TL/DD/9122-2

FIGURE 1. Port PC Latched Address Output, PRLSEL, and $\overline{\text{RESET}}$ Timing



TL/DD/9122-3

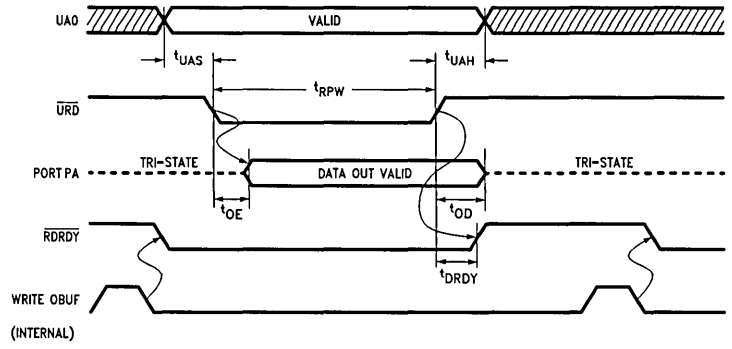
FIGURE 2. Read Cycle



TL/DD/9122-4

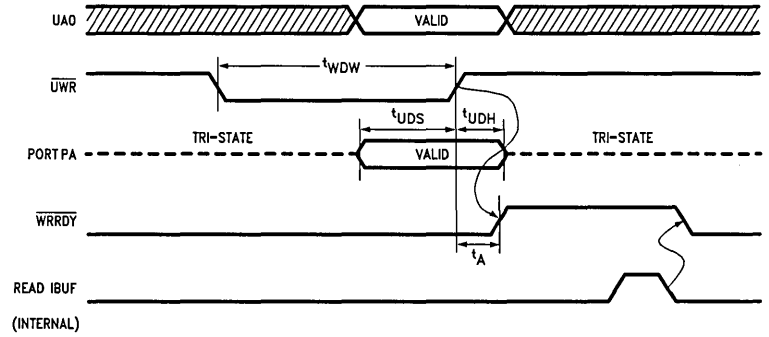
FIGURE 3. Write Cycle

Timing Waveforms (Continued)



TL/DD/9122-5

FIGURE 4. UPI Read Timing



TL/DD/9122-6

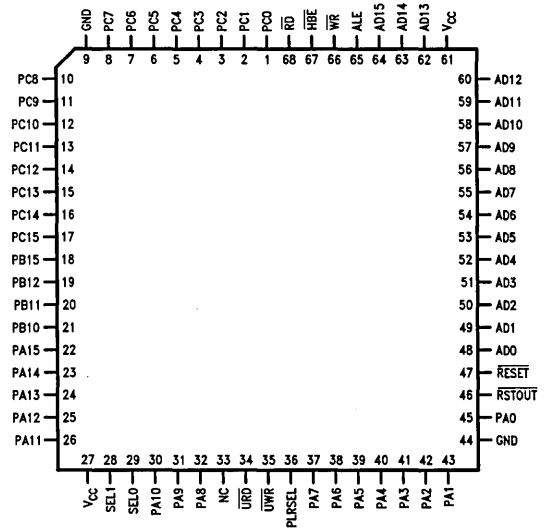
FIGURE 5. UPI Write Timing

Pin Descriptions

Pin	Description
V_{CC} (2)	Supply voltage
GND (2)	Ground reference
$\overline{RESET}$	Chip reset (active low). Schmitt trigger input which initializes PEARL and will TRI-STATE® all ports
$\overline{RSTOUT}$	Reset Output (active low) which can be used to reset the HPC and any other PEARL chips in the same system
PRLSEL	An output (high assert) signalling when the address on the AD port has selected a PEARL register for that particular PEARL configuration (i.e., PEARL 0, PEARL 1, PEARL 2, PEARL 3). This output is useful for disabling memory data which may reside at the same addresses as the PEARL registers.
AD0–AD15	16-bit multiplexed address/data bus
ALE	Address Latch Enable input
$\overline{WR}$	Write Input
$\overline{HBE}$	High Byte Enable Input
$\overline{RD}$	Read Input
PA0–PA15	16-bit bidirectional input/output port
PB10/UA0	Output only pin, or UA0 input in UPI mode
PB11/$\overline{WRRDY}$	Output only pin, or $\overline{WRRDY}$ output in UPI mode
PB12	Output only pin
PB15/$\overline{RDRDY}$	Output only pin, or $\overline{RDRDY}$ output in UPI mode
PC0–PC15	Latched external address bits (outputs), or 16-bit I/O port (bidirectional). Port PC is a latched version of the address output on the AD0–AD15 bus if SEL0 and SEL1 are both low and a 16-bit bidirectional I/O port otherwise.
SEL1,0	Two inputs which specify the PEARL number of the port expander (00 = PEARL 0, 01 = PEARL 1, 10 = PEARL 2, 11 = PEARL 3)
$\overline{URD}$	UPI read strobe (input, low assert) which causes the PEARL to output OBUF (UPI output buffer) on the PA bus if the PEARL is in UPI mode. When not using the PEARL 0-UPI mode, this input should be tied to V_{CC}.
$\overline{UWR}$	UPI write strobe (input, low assert) which causes the PEARL to latch the data present on the PA bus into IBUF (the UPI input buffer) if the PEARL is in UPI mode. When not using the PEARL 0-UPI mode, this input should be tied to V_{CC}.
NC	No Connect

Connection Diagram

Plastic Chip Carrier



TL/DD/9122-7

Top View

Order Number HPC46900V
See NS Package Number V68A

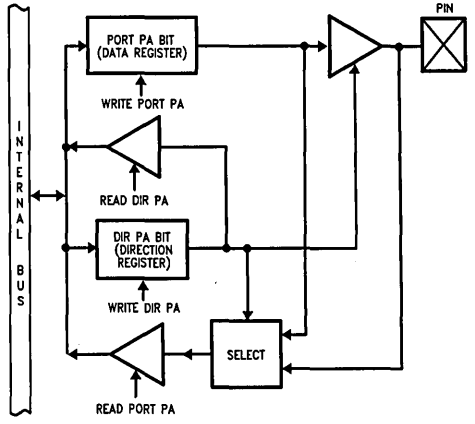
Ports PA, PB, and PC

The highly flexible PA, PB, and PC ports are similarly structured. Port PA (see *Figure 6*) and Port PC (see *Figure 7*) consist of a data register and a direction register. Port PB (see *Figure 8*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. A port pin is selected as an input and placed in a TRI-STATE mode by clearing the corresponding bit in the direction register.

A write operation to a port pin configured as an input causes the value to be written into the data register. A read operation returns the value detected at the pin. Writing to a port pin configured as an output writes the value into the data register and causes the pin to output the same value. Reading a port pin configured as an output returns the value held in the data register.

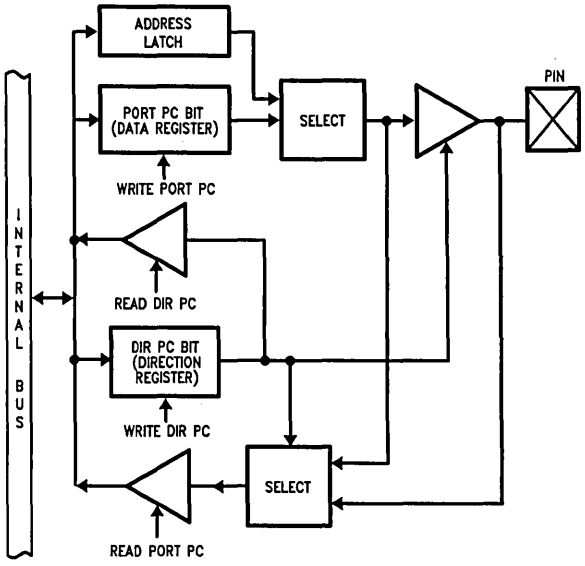
Primary and secondary functions are multiplexed onto Port PB through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.



TL/DD/9122-8

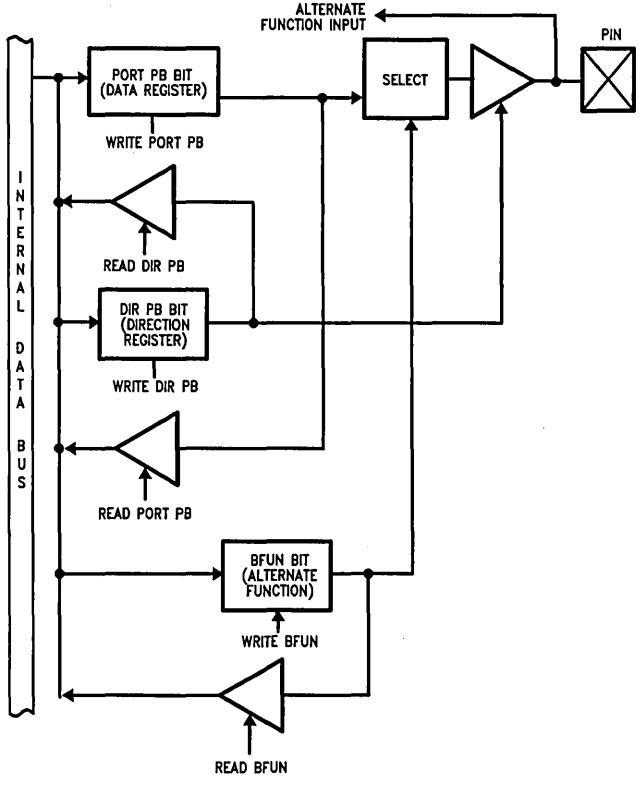
FIGURE 6. Port PA I/O Structure

Ports PA, PB, and PC (Continued)



TL/DD/9122-9

FIGURE 7. Port PC Structure: I/O and Latched Address



TL/DD/9122-10

FIGURE 8. Structure of Port PB Pins PB10, PB11, PB12, PB15

Operating Modes

TABLE I. PEARL Functions

	Inputs				Port Functions						
PEARL Mode	**SEL1**	**SEL0**	**UPIEN**	**8OR16**	**PA0–PA7**	**PA8–PA15**	**PB10**	**PB11**	**PB12**	**PB15**	**PC0–PC15**
PEARL 0—I/O	0	0	0	X	I/O	I/O	Out	Out	Out	Out	ADDR
PEARL 0—UPI (16)	0	0	1	0	UPI BUS	UPI BUS	UA0	$\overline{\text{WRRDY}}$*	Out	$\overline{\text{RDRDY}}$*	ADDR
PEARL 0—UPI (8)	0	0	1	1	UPI BUS	I/O	UA0	$\overline{\text{WRRDY}}$*	Out	$\overline{\text{RDRDY}}$*	ADDR
PEARL 1	0	1	0	X	I/O	I/O	Out	Out	Out	Out	I/O
PEARL 2	1	0	0	X	I/O	I/O	Out	Out	Out	Out	I/O
PEARL 3	1	1	0	X	I/O	I/O	Out	Out	Out	Out	I/O

*If corresponding bit in BFUN Register is set.

The two inputs, SEL0 and SEL1, along with the bits UPIEN and UPI8BIT in the UPIC register determine the function of the PEARL as described below and summarized in Table 1.

When interfacing the PEARL to an HPC microcontroller, the microcontroller must be in 16-bit mode.

PEARL 0—I/O

In this mode, ports PA and PB are memory mapped I/O, and port PC is a latched address output from the multiplexed address/data bus, AD0–AD15. The host HPC must be either Expanded Normal mode (EA bit in the PSW = "1", EXM = "0"), or Expanded ROMless mode (EA bit in the PSW = "1", EXM = "1". *Figure 9* shows a HPC in Expanded ROMless mode with the address range 200–FFFF being addressed through the PEARL.

PEARL 0—UPI

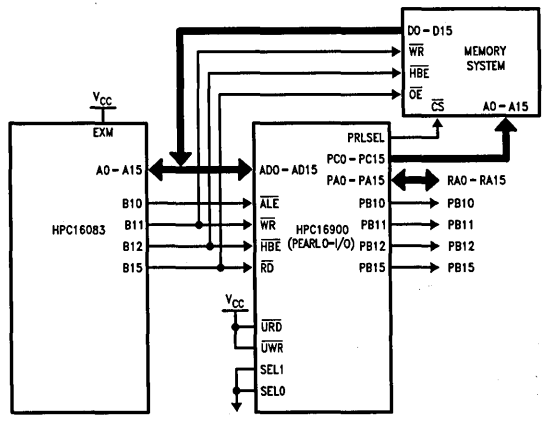
Port PA is either a 16-bit UPI data bus, or an 8-bit UPI data bus on the lower bytes and I/O on the upper bytes. Of the four PB pins, three are UPI control signals, and one is a programmable output. The PC port is a latched address output from the multiplexed address/data bus, AD0–AD15. The host HPC must have memory in the address range 200–FFFF addressed through the PEARL. When using a PEARL 0—UPI with an HPC, the HPC must be in the Expanded ROMless mode (EA bit in the PSW = "1", EXM = "1") as shown in *Figure 10*.

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows a system with a HPC160xx and a PEARL 0 configured for UPI operation to be used as an intelligent peripheral to another processor.

The interface consists of a UPI Data Bus (Port PA), a Read Strobe ($\overline{\text{URD}}$), a Write Strobe ($\overline{\text{UWR}}$), a Read Ready Line ($\overline{\text{RDRDY}}$), a Write Ready Line ($\overline{\text{WRRDY}}$) and one address input (UA0). The UPI Data Bus can be either eight or sixteen bits wide. The $\overline{\text{URD}}$ and $\overline{\text{UWR}}$ inputs, and the $\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$ outputs may be used to interrupt the host processor as shown in *Figure 10*.

The registers controlling the UPI logic are the Input Buffer (IBUF), Output Buffer (OBUF) and a Control Register (UPIC).



TL/DD/9122–11

FIGURE 9. PEARL 0—I/O with External Memory

Operating Modes (Continued)

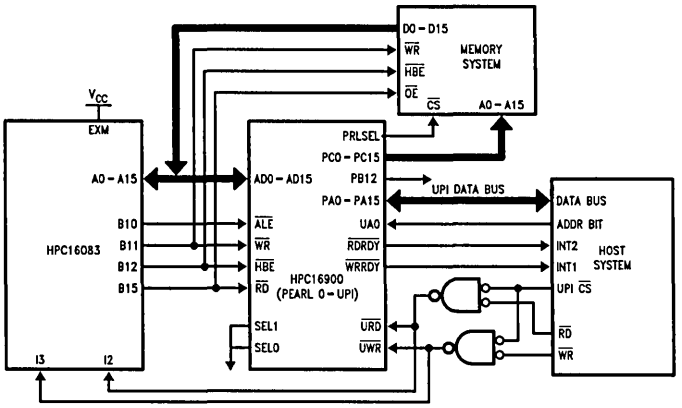
Refer to the HPC Users' Manual for a detailed functional description of the UPI mode.

PEARL 1, 2, 3

Each one of these configurations provides memory mapped I/O on all three ports. Ports PA and PC each provide 16 bits of I/O, while PB is output only. *Figure 11* shows a PEARL 1 and a PEARL 2 configured as port expanders. When using a PEARL 1, 2, or 3 with a HPC, the HPC may be configured in the Expanded Normal mode or the Expanded ROMless mode.

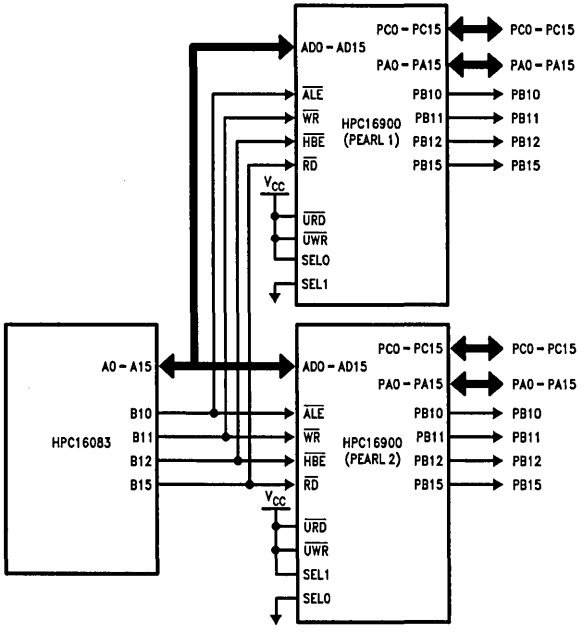
HPC Emulation

A system using a PEARL configured as a PEARL 0—I/O, a HPC in Expanded ROMless mode, and an EPROM *(Figure 9)* can be considered a pseudo emulator of a HPC in Single Chip Normal mode. In this configuration, the PEARL re-creates the I/O functionality of HPC's Port A and four bits of Port B which are used to address off chip memory. The only difference between the system shown in *Figure 9* and a mask programmed HPC in Single Chip Normal mode is that the registers associated with Port PA and Port PB of the PEARL are at different locations than the registers of the HPC's Port A and Port B (see PEARL Register Address Assignments).



TL/DD/9122-12

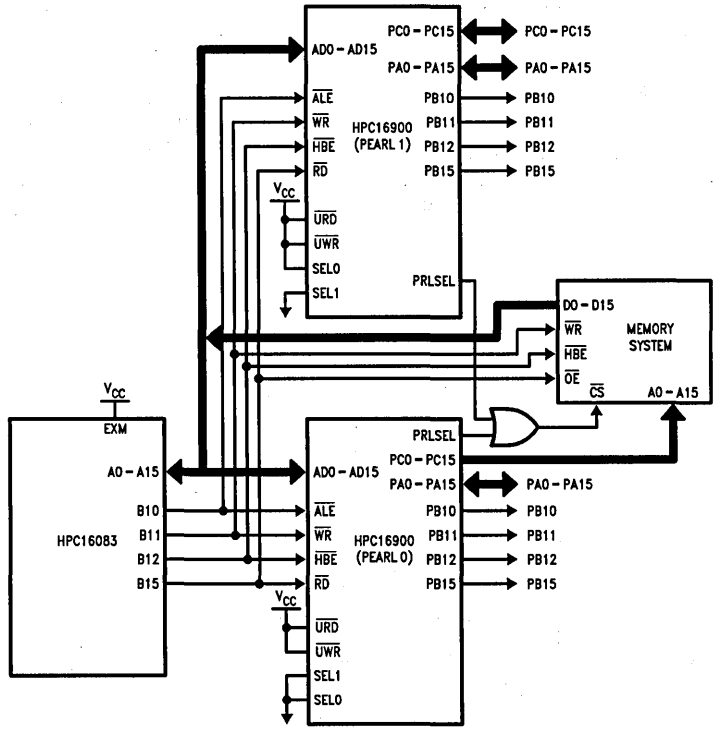
FIGURE 10. PEARL 0—UPI (16-bit) with External Memory



TL/DD/9122-13

FIGURE 11. PEARL 1 and PEARL 2 Configured as Port Expanders

PRLSEL Operation



TL/DD/9122–14

FIGURE 12. PEARL 0 and PEARL 1 with External Memory

This configuration, however, will not emulate a HPC in Expanded Normal mode since the PEARL's Port PA and PB will not emulate the address/data bus function.

The PRLSEL output is asserted when the address presented on the address data bus (AD0–AD15) corresponds to a register in the PEARL's address block (see PEARL Register Address Assignments). For example, an HPC16900 configured as a PEARL 0 will assert PRLSEL only if the address is within the PEARL 0 address block. Likewise, a PEARL 1 will assert its PRLSEL when the address is within the PEARL 1 address block.

Therefore, when using multiple PEARLs in a system, the PRLSEL outputs must be ORed together as in *Figure 12*. In this system, the PEARL addresses overlap those of the Memory System, and the ORed PRLSEL signal is used to deselect the outputs of the Memory System to avoid bus contention.

PRLSEL is an output derived from the internal address decode logic, which decodes the AD0–AD15 inputs while ALE is high. During the time ALE is high, the state of PRLSEL is indeterminate.

On the falling edge of ALE, the decoding is stopped, and PRLSEL is latched to a valid state. This state is guaranteed to be valid until ALE goes high during the next address cycle.

Initialization

Immediately following RESET the PEARL will be in the following state:

I/O AND OUTPUT ONLY PINS

AD0–15	TRI-STATE
PB10, 11, 12, 15	TRI-STATE
PA0–PA15	TRI-STATE
PC0–PC15	TRI-STATE if PEARL 1, PEARL 2, or PEARL 3
	PEARL 0: Power on Reset—indeterminate. Otherwise, asserted, bits 0–15 of most recent latched address.
PRLSEL	Power on Reset—indeterminate until the first ALE will put PRLSEL in a known state based on the first address output.
	Otherwise, asserted (may be high or low). If the address of one of the PEARL's registers was the most recent address received by the PEARL, then PRLSEL will be high, otherwise PRLSEL will be low.

Initialization (Continued)

INTERNAL REGISTERS AND LATCHES

Port PA Data Register	Indeterminate
Port PA Direction Register	Cleared
Port PB Data Register	Indeterminate
Port PB Direction Register	Cleared
Port PB Function Register	Cleared
Port PC Data Register	Indeterminate
Port PC Direction Register	Cleared
UPIENB Latch	Cleared
UPI8BIT	Cleared

PEARL REGISTER ADDRESS ASSIGNMENTS

The following registers have been assigned to the block of addresses from hex 800 to 83F:

800, 801	Reserved
802, 803	PEARL 0 Port PA Data/UPI Output Buffer
804, 805	PEARL 0 Port PA Direction/UPI Input Buffer
806, 807	PEARL 0 Port PB Data (4 bits only)
808, 809	PEARL 0 Port PB Direction (4 bits only)
80A, 80B	PEARL 0 Port PC Data
80C, 80D	PEARL 0 Port PC Direction
80E, 80F	Reserved
810, 811	Reserved
812, 813	PEARL 1 Port PA Data
814, 815	PEARL 1 Port PA Direction
816, 817	PEARL 1 Port PB Data (4 bits only)
818, 819	PEARL 1 Port PB Direction (4 bits only)
81A, 81B	PEARL 1 Port PC Data
81C, 81D	PEARL 1 Port PC Direction
81E, 81F	Reserved
820, 821	Reserved
822, 823	PEARL 2 Port PA Data
824, 825	PEARL 2 Port PA Direction
826, 827	PEARL 2 Port PB Data (4 bits only)
828, 829	PEARL 2 Port PB Direction (4 bits only)
82A, 82B	PEARL 2 Port PC Data
82C, 82D	PEARL 2 Port PC Direction
82E, 82F	Reserved
830, 831	Reserved
832, 833	PEARL 3 Port PA Data
834, 835	PEARL 3 Port PA Direction
836, 837	PEARL 3 Port PB Data (4 bits only)
838, 839	PEARL 3 Port PB Direction (4 bits only)
83A, 83B	PEARL 3 Port PC Data
83C, 83D	PEARL 3 Port PC Direction
83E, 83F	Reserved

When the PEARL is configured as a PEARL 0—UPI, these register locations are used in UPI operation.

UPIC is accessed at E7, E6

BFUN is accessed at F5, F4

Ordering Information

The following chart explains how the various options are designated in the part number.

HPC16900V

- Package Type
  - V = Plastic Leaded Chip Carrier (PLCC)
- Temperature
  - 4 = Commercial (0°C TO +70°C)
  - 3 = Industrial (−40°C TO +85°C)
  - 2 = Automotive (−40°C TO +105°C)
  - 1 = Military (−55°C TO +125°C)

TL/DD/9122-15

Section 5

HPC Applications

Section 5 Contents

HPC MICROWIRE/PLUS™ Master-Slave Handshaking Protocol

National Semiconductor
Application Note 474
Richard Lazovick

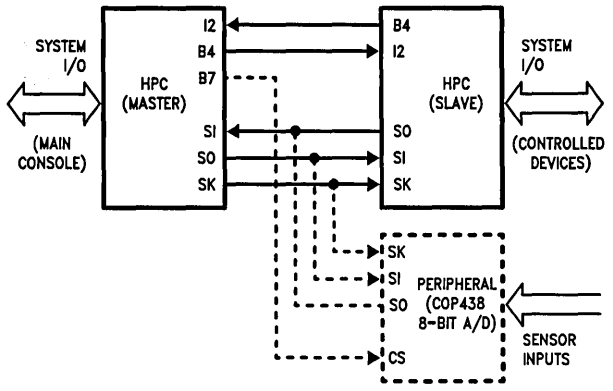
INTRODUCTION

This applications note describes how to use National Semiconductor's MICROWIRE/PLUS to communicate between two members of the HPC family of microcontrollers, and will discuss the implications of adding other MICROWIRE™ peripherals. MICROWIRE/PLUS (μWIRE) may be effectively used to communicate between chips, such as in Small Area Networks (SANs). Possible applications range from setting up a communications network within an automobile to home security systems. Among the standard MICROWIRE peripherals available are display drivers (LCD, VF, LED), memories (RAM, EEPROM), A/D converters, and frequency generators/timers. Each MICROWIRE peripheral requires its own handshaking protocol, however the HPC's MICROWIRE is flexible enough to work with any peripheral and allows you to define your own handshaking protocol when having two HPC family members communicate.

MICROWIRE

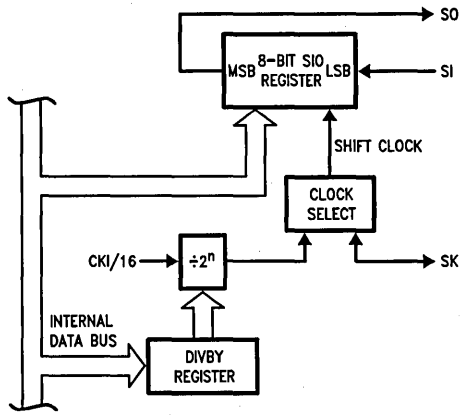
MICROWIRE/PLUS is an extension of National Semiconductor's MICROWIRE communications interface. It allows high speed two way serial communications between a master processor and one or more slave processors or peripherals. MICROWIRE/PLUS uses only three wires plus chip selects, therefore it saves on intricate bus routing and does not waste 8-bit ports. *Figure 1* shows the block diagram of a sample application using two HPC family members and an 8-bit A/D peripheral to monitor and control certain environmental conditions within a system.

MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register (SIO) using SI as the serial input and SO as the serial output. The contents of the SIO register may be accessed through any of the memory access instructions. SK is the clock for the SIO register (see *Figure 2*). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. Data to be transmitted from the SIO register is shifted out on the falling edge of the SK clock. Serial data on the SI pin is latched in on the rising edge of the SK clock (see *Figure 3* μWIRE Timing).



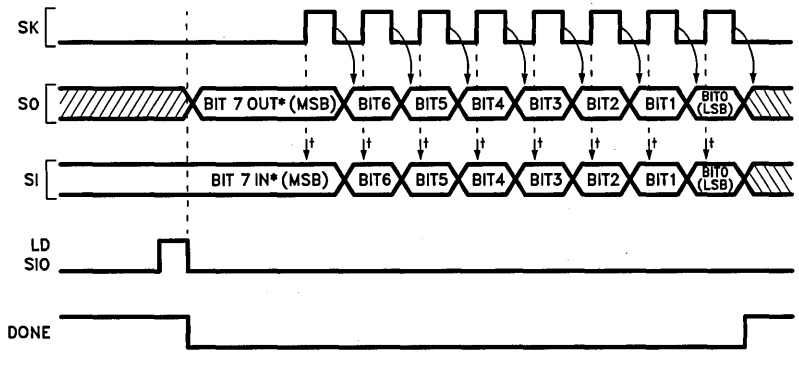
TL/DD/9140–1

FIGURE 1. HPC μWIRE Block Diagram (Environmental Control System)



Note: The most significant bit is shifted out first. The SO pin reflects the contents of the MSB in the SIO register.

FIGURE 2. MICROWIRE/PLUS Block Diagram



Note: The first bit of every eight bits in the SIO register being shifted out will have a longer duration then the other bits. This results from the hardware implementation used for MICROWIRE.

* This bit becomes valid immediately when the transmitting device loads its SIO register.

† Arrows indicate points at which SI is sampled.

FIGURE 3. μWIRE Timing

A μWDONE flag in the IRPD (Interrupt Pending) register indicates when the data shift is completed.

The HPC can enter the MICROWIRE/PLUS mode as a master or a slave. The μWMODE control bit in the IRCD (Interrupt Condition) register determines whether the HPC is a master or slave. The shift clock is generated internally when the HPC is configured as a master. An externally generated shift clock on the SK pin is used when the HPC is configured as a slave. When the HPC is a master, the DIVBY register allows the SK clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz when CKI is 16 MHz (see Table I).

HOW TO USE MICROWIRE/PLUS

To use MICROWIRE, start by setting up the B port appropriately for the MICROWIRE functions. The SO and SK functions are multiplexed onto Port B pins B5 and B6 respectively. For the master, set bits 5 and 6 in the DIRB register (direction register for Port B) to set SO and SK as outputs. For the slave, set bit 5 and reset bit 6 in the DIRB register to set SO as an output and SK as an input. The BFUN register (Port B function register) is used to set SO and SK as alternate functions in the master and only SO as an alternate function in the slave. The MICROWIRE/PLUS mode can be enabled or disabled any time under program control. This is done through the BFUN register. Placing a "1" in the corresponding bit location causes the alternate function to be activated, a "0" causes the alternate function to be disabled. It is good practice to initialize the output pins by setting PORTB (Port B data register) to a known state.

The SI function is multiplexed onto Port I pin I5. This pin is always an input and the SI function is automatically selected when in the MICROWIRE mode. Setting the μWMODE control bit, bit 1, in the IRCD register will enable the part to be a master, resetting the bit will make it a slave. For the master, the DIVBY register has to be initialized to set the appropriate SK frequency (see Table I.). For example if the crystal frequency is 16 MHz and an SK frequency of 1 MHz is desired, load the least significant nibble of the DIVBY register with 2 (16 MHz/16 = 1 MHz).

For a summary of the register and pin configurations for the master and slave modes see Table II.

TABLE I. HPC μWIRE DIVBY Register

μWIRE SK Divisor				
MSB			**LSB**	**CLOCK**
0	0	0	0	not allowed
0	0	0	1	not recommended*
0	0	1	0	CKI/16
0	0	1	1	CKI/32
0	1	0	0	CKI/64
0	1	0	1	CKI/128
0	1	1	0	CKI/256
0	1	1	1	CKI/512
1	0	0	0	CKI/1024
1	0	0	1	CKI/2048
1	0	1	0	CKI/4096
1	0	1	1	CKI/8192
1	1	0	0	CKI/16384
1	1	0	1	CKI/32768
1	1	1	0	CKI/65536
1	1	1	1	CKI/131072

*This option uses timer T3 output, but does not generate a square wave. (See HPC users manual for more details.)

TABLE II. μWIRE Register and Pin Conditions for Master and Slave Operation

Operation	μWMODE bit	BFUN B5	BFUN B6	DIRB B5	DIRB B6	PIN B5	PIN B6	PIN I5
MICROWIRE Master	1	1	1	1	1	SO	INT. SK	SI
MICROWIRE Master	1	1	1	0	1	TRI-STATE®	INT. SK	SI
MICROWIRE Slave	0	1	0	1	0	SO	EXT. SK	SI
MICROWIRE Slave	0	1	0	0	0	TRI-STATE	EXT. SK	SI

DEFINING THE MASTER/SLAVE HANDSHAKING PROTOCOL

There are a few things to keep in mind when defining a handshaking protocol for the HPC:

1) Only the master can generate SK clocks.
2) As 8 bits are shifted into the SIO register, the 8 bits already in there are shifted out.
3) After 8 bits are shifted into (or out of) the SIO register the MICROWIRE done (μWIRE DONE) flag gets set.
4) ANY access to the SIO register in the master that performs a write operation causes the contents of SIO to be shifted out.
5) No data will be shifted into or out of the slave's SIO register if its μWIRE DONE flag is set.
6) Any write to the SIO register in the master or slave resets its μWIRE DONE flag.

Keeping the above six points in mind, let's look at one possible handshaking protocol between a master HPC and a slave HPC. Number two above tells us we can send and receive data at the same time, however since only the master initiates data transfer we want to be sure the slave is ready before we get started with the exchange. Since the master initiates the transfer process there is no need for the master's MICROWIRE routine to be interrupt driven (though it can be if it is desired to have the slave initiate data transfers also). On the other hand, since the slave will be off doing other tasks it is most effective to have its MICROWIRE routine be interrupt driven.

A FEW THINGS TO NOTE ABOUT THE PROGRAMS

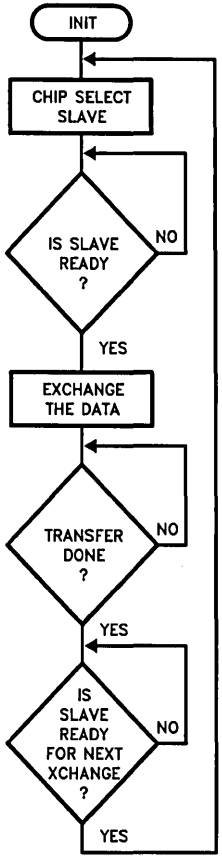
The following programs refer to the system configuration shown in *Figure 1.* This example code does a simple data transfer. The master reads in data on Port D, sends it via MICROWIRE to the slave, and reads it back. They both start by initializing the chip mode and number of wait states (PSW), disabling interrupts, setting the DIVBY register as necessary, initializing Port B, and enabling the appropriate MICROWIRE mode (IRCD). Then the slave continues with its main code (a wait loop) until interrupted. When the master decides it's ready to send MICROWIRE data, it signals the slave by setting the slave interrupt pin on Port B, then it waits for the slave to respond.

Meanwhile, the slave goes into action. It clears the μWDONE flag and loads the SIO register (X A, SIO), then notifies the master that it is ready to continue. Once the master realizes the slave is ready to continue, it removes the interrupt signal to the slave (RESET PORTB.SLAVI), reads in the data to be sent (LD A, PORTD), and starts transmitting it (X A, SIO). At the same time the master reads in the data received at the last data exchange with the slave. Then the master loops until it is done transferring data and loops again until the slave is finished with its interrupt routine. In a real program the master would be off executing code and not having to wait in these loops. Once the transmission is complete the slave reads in the new data (LD A, SIO), lets the master know it is done with its interrupt routine (RESET PORTB.MASTR), and re-enables interrupts as it returns to the main routine (RETI).

In the master's code there is only one access to the SIO register and that access is an exchange. Remember point #4, we can take advantage of the exchange instruction (X A, SIO), which is a read-modify-write instruction. Therefore, with one instruction, we can read the data from the previous transfer into the accumulator, and write the data to be transferred into the SIO register. If this method is not practical, then separate read and write instructions must be used.

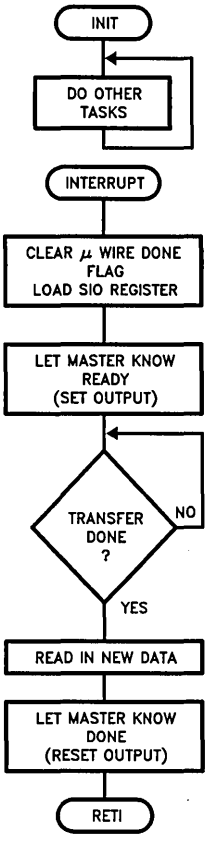
When accessing the SIO register be sure the μWIRE DONE flag is set so you know the data is not changing. At other times we have to be sure the flag is reset or no data will ever be transferred (shifted in or out). Notice that the "X A, SIO" was used to reset the μWIRE DONE flag as well as load the register with the data to be sent.

MASTER'S Flow Chart



TL/DD/9140-4

SLAVE'S Flow Chart



TL/DD/9140-5

MASTER's SAMPLE CODE

```
;
;VARIABLE DECLARE
;

PSW    = M(00C0)

BFUN   = W(0F4)            ;Port B ALTERNATE FUNCTION REGISTER
DIRB   = W(0F2)            ;Port B DIRECTION REGISTER
PORTB  = W(0E2)            ;Port B DATA REGISTER
PORTD  = M(0104)           ;Port D (INPUT PORT)

ENIR   = M(0D0)            ;INTERRUPT ENABLE REGISTER
IRPD   = M(0D2)            ;INTERRUPT PENDING REGISTER
IRCD   = M(0D4)            ;INTERRUPT CONDITION REGISTER
SIO    = M(0D6)            ;SERIAL I/O REGISTER
PORTI  = M(0D8)            ;INTERRUPT (AND uWIRE SERIAL IN) INPUT PORT
DIVBY  = W(018E)           ;TIMER DIVIDE BY REGISTER

SLAVI  = 4                 ;SLAVE INTERRUPT BIT (IN Port B)
uWDONE = 0                 ;uWIRE DONE BIT (IN IRPD)
uWMODE = 1                 ;uWIRE MASTER/SLAVE BIT (IN IRCD)
SK     = 6                 ;uWIRE SERIAL CLOCK (IN Port B)
SLAVR  = 2                 ;SLAVE RESPONSE BIT (IN Port B)
```

MASTER's SAMPLE CODE (Continued)

```
.=0F800                 ;START PROGRAM

BEGIN:
        LD      PSW,008             ;SINGLE CHIP MODE, 1 WAIT STATE
        LD      ENIR,00             ;DISABLE ALL INTERRUPTS
        LD      DIVBY,02222         ;uWIRE CLOCK = /16

        LD      DIRB,0FFFF          ;Port B ALL OUTPUTS
        LD      BFUN,00060          ;ONLY SO & SK HAVE ALTERNATE FUNCTIONS
        LD      PORTB,00000         ;INIT PORTB TO ALL ZEROs

        SET     IRCD.uWMODE         ;SET THIS HPC AS MASTER

DOITAG:                         ;JUMP TO HERE TO DO IT AGAIN
        SET     PORTB.SLAVI         ;NOTIFY SLAVE (INTERRUPT THE SLAVE)

WAIT:
        IF      PORTI.SLAVR         ;SLAVE READY?
        JP      SLAVRS              ;GO SEND/RECEIVE uWIRE DATA
        JP      WAIT                ;NO IT IS NOT READY YET

SLAVRS:
        RESET PORTB.SLAVI           ;REMOVE SLAVE NOTIFIER
        LD      A,PORTD             ;LOAD A W/ DATA TO SEND
        X       A,SIO               ;SEND NEW DATA AND READ DATA FROM
                                    ;...LAST uWIRE EXCHANGE

DONE:
        IF      IRPD.uWDONE         ;WAIT TILL DONE EXCHANGING
        JP      CONT                ;uWIRE IS DONE
        JP      DONE                ;uWIRE NOT DONE (KEEP TESTING)
CONT:
        IF      PORTI.SLAVR         ;IS SLAVE READY TO CONTINUE?
        JP      CONT                ;NO
        JP      DOITAG              ;START ALL OVER (DO IT AGAIN)

.END BEGIN
```

SLAVE's SAMPLE CODE

```
;
;VARIABLE DECLARE
;

PSW     = M(00C0)

BFUN    = W(0F4)            ;Port B ALTERNATE FUNCTION REGISTER
DIRB    = W(0F2)            ;Port B DIRECTION REGISTER
PORTB   = W(0E2)            ;Port B DATA REGISTER
```

SLAVE's SAMPLE CODE (Continued)

```
ENIR   = M(0D0)              ;INTERRUPT ENABLE REGISTER
IRPD   = M(0D2)              ;INTERRUPT PENDING REGISTER
IRCD   = M(0D4)              ;INTERRUPT CONDITION REGISTER
SIO    = M(0D6)              ;SERIAL I/O REGISTER
SO     = 5                   ;uWIRE SERIAL OUTPUT PIN (ON Port B)
MASTR  = 4                   ;MASTER RESPONSE BIT (IN Port B)
uWDONE = 0                   ;uWIRE DONE BIT (IN IRPD)
uWMODE = 1                   ;uWIRE MASTER/SLAVE BIT (IN IRCD)
INT2   = 2                   ;INTERRUPT 2 BIT

.=0FFFA                      ;INT2 - INTERRUPT VECTOR
.WORD MASNOT                 ;...MASTER NOTIFICATION
.=0F800                ;START PROGRAM

BEGIN:
       LD     PSW,008            ;SINGLE CHIP MODE, 1 WAIT STATE
       LD     ENIR,01            ;DISABLE ALL INTERRUPTs, BUT ENABLE GIE

       LD     DIRB,0FF10         ;Port B UPPER, & MASTR ARE OUTPUTS
                                 ;...(use LD DIRB,0FF30 to set SO as an
                                 ;...output if not using any peripherals)
       LD     BFUN,00020         ;ONLY SO HAS ALTERNATE FUNCTION
                                 ;...NOTE: SK is NOT an alternate
                                 ;...function in the slave!
       LD     PORT B,00000       ;INIT PORTB TO ALL ZEROS

       RESET  IRCD.uWMODE        ;SET THIS HPC AS A SLAVE
       SET    IRCD.INT2          ;SET INT2 INTERRUPT (+) POLARITY
       SET    ENIR.INT2          ;ENABLE EXTERNAL INTERRUPT TO
                                 ;...RECEIVE SLAVE RESPONSE

PAU:
       JP     PAU                ;WAIT HERE FOR INTERRUPT FROM MASTER

MASNOT:                ;uWIRE INTERRUPT ROUTINE

       X      A,SIO              ;CLEAR uWDONE FLAG (AND LOAD DATA FROM
                                 ;...ACCUMULATOR TO SEND)
       SET    PORTB.SO           ;ENABLE SO (needed only if using a peripheral)
       SET    PORTB.MASTR        ;NOTIFY MASTER THAT READY TO CONTINUE

NOTDN:
       IF     IRPD.uWDONE        ;WAIT TILL DONE SHIFTING
       JP     DONE               ;DONE, GO CONTINUE
       JP     NOTDN              ;NOT DONE, CONTINUE LOOPING

DONE:
       LD     A,SIO              ;READ IN NEW DATA

       RESET  PORT B.SO          ;TRI-STATE SO (needed only if
                                 ;              using a peripheral)
       RESET  PORTB.MASTR        ;REMOVE SIGNAL TO MASTER
       RETI

.END BEGIN
```

ADDING PERIPHERALS OR ANOTHER SLAVE

Adding another slave HPC or a peripheral to the above Microwire configuration can add more power to your design with minimal extra cost and design time. In *Figure 1,* an extra peripheral is shown in dotted outline form. The hardware and software modifications are straightforward, however there are a few considerations to keep in mind:

— Tri-state the SO pin on the slave HPC by resetting B5 in the DIRB register when the slave is not 'chip-selected' by the master.

— When adding more HPC slaves, the master's and slave's routines remain the same. Only different B port pins for chip select and I or B port pins for slave acknowledge need to be used.

— For peripherals the principals of operation are still the same and so are the initialization procedures, however some of the code will have to be modified to accommodate the specific handshaking required by the peripheral. (Note: some of the peripherals require 16 or more consecutive bits without interruption of the SK clock. To provide continuous SK clocks, set up the accumulator with next byte of data to send, loop until μWDONE is set, then exchange the contents of the accumulator and the SIO register (X A, SIO). The above steps will provide nearly continuous SK clocks—the slower the SK clock is set for, the more continuous they will appear.)

APPLICATIONS

Now that you are more familiar with MICROWIRE/PLUS, where can you get experience using it?

— It can be used in a security system where the on-site master lets the periphery slaves know which security codes they can now let in, while at the same time the slaves monitor fire alarms and smoke detectors.

— It can be used in automotive brakes to allow all the wheels to communicate with each other. The wheels can trade information on road conditions and a master can monitor all four wheels to coordinate them and check for malfunctions.

— It can be used in a robot arm to allow each joint to make the decision as to how it will help the entire arm reach its final position. This application is one example of how MICROWIRE/PLUS can be used for system task partitioning.

— It can be used in a MUX-WIRING system.

When using MICROWIRE to communicate between two chips on the same board, a high data rate can be used. When communicating over longer distances, slower speeds should be used.

SUMMARY

MICROWIRE/PLUS can be a very powerful tool that can easily add power to a microcontroller based system. It is easy to use and does not require much hardware to implement. To add a new feature to your current design, choose a peripheral and add a small amount of code. To start using MICROWIRE, define the handshake protocol best suited for your application keeping in mind the six points given above in the 'Defining the Master/Slave Handshaking Protocol' section. Then initialize the appropriate registers: BFUN, DIRB, PORTB, DIVBY, and IRCD. The MICROWIRE circuitry will then run independent of the CPU except to exchange data between the SIO register and the CPU, and to initiate the data exchange between the master and slaves. With a CPU clock of 16 MHz, MICROWIRE/PLUS may achieve a maximum data rate of 1 MHz. MICROWIRE can be used to add display controllers, A/D's, memories, timers, and even other microcontrollers to an HPC microcontroller based design. Remember MICROWIRE/PLUS is not a trivial piece of very fine wire, it is a high speed two way serial communications interface!

Interfacing Analog Audio Bandwidth Signals to the HPC

National Semiconductor
Application Note 484
Ashok Krishnamurthy

INTRODUCTION

This report describes a method of interfacing analog audio bandwidth signals to the National Semiconductor HPC microcontroller. The analog signal is converted to a digital value using the National Semiconductor TP3054 codec/filter combo. The digital value is then transferred to the HPC using the MICROWIRE/PLUS™ synchronous serial interface. The digital output sample computed by the HPC is also transferred to the TP3054 using the MICROWIRE/PLUS interface. The TP3054 then converts this digital value to an analog signal.

ADVANTAGES OF USING A CODEC

There are a number of advantages in using a codec for A/D and D/A conversion of analog signals.

1. The codec/filter combos such as the TP3054 integrate a number of functions on a single chip. Thus the TP3054 includes the analog anti-aliasing filters, the Sample-and-Hold circuitry and the A/D and D/A converters for analog signal interfacing.
2. The μ-law coding effectively codes a 14-bit conversion accuracy in 8 bits. This allows the interface to the HPC to be greatly simplified.

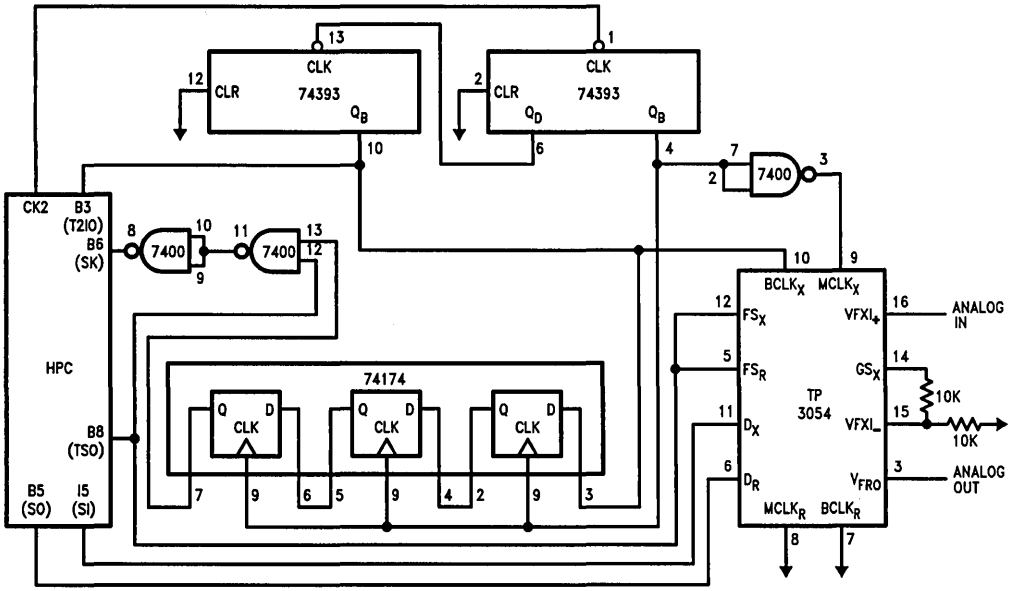
DISADVANTAGES IN USING A CODEC

While the use of a codec is appropriate for audio (in particular speech) processing applications, it has a number of disadvantages in other cases.

1. The sampling rate is fixed at 8 kHz. If lower or higher sampling rates are desired, the codec cannot be used. Note that the real-time signal processing that the HPC can perform at a 8 kHz sampling rate is limited.
2. The resolution is fixed, and is about 14 bits/sample.
3. Digital filtering algorithms require that the samples used in the processing be linear coded PCM. Thus the 8-bit μ-law PCM values output by the codec need to be converted to linear coded PCM. Correspondingly, the output of the digital filter, which is in linear coded PCM needs to be converted to 8-bit μ-law PCM before outputting to the codec. This requires additional processing per sample.

DESCRIPTION OF THE INTERFACE

The circuit schematic of the interface is shown in *Figure 1*. Note that the schematic does not show complete details of the HPC. Only the HPC pins that are relevant to this interface are shown. A wire-wrapped version of the circuit has been constructed on a NSC HPC 16040 Chip Carrier Board.



TL/DD/9246–1

FIGURE 1. Circuit Schematic

Note that this report does not go into the details about the use of the TP3054 codec chip or programming the HPC. It also does not discuss the μ-law to linear and linear to μ-law code conversion in detail. For more information on these issues, please consult the references listed at the end.

1. **Codec Signalling Considerations.** The TP3054 can operate in either synchronous or asynchronous modes. Further, in each of these modes, it uses short or long frame sync operation. The circuit shown in *Figure 1* runs the codec in synchronous mode with long-frame-sync operation.

The codec requires 4 clock sources for proper operation in the synchronous mode. These are MCLK-x, BCLK-x, FS-x and FS-r. MCLK-x is a master clock and is used to clock the switched-capacitor filters. BCLK-x is the bit shift clock, and FS-x and FS-r are the frame sync clocks. These clocks need to be synchronous.

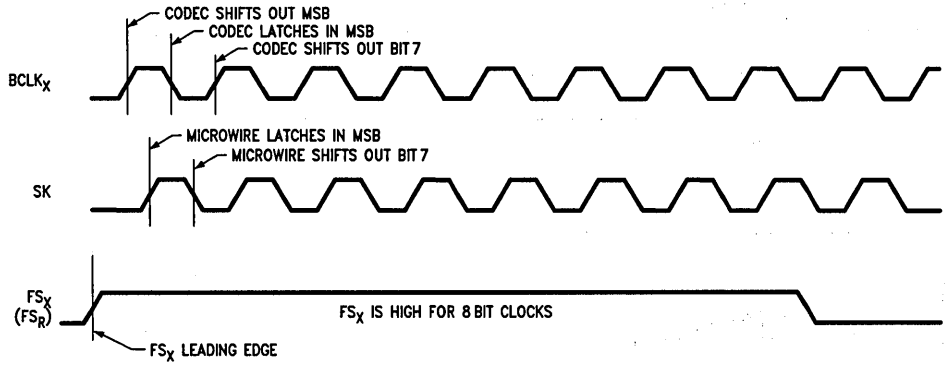
These clocks are obtained in the circuit as follows. MCLK-x is obtained by dividing the HPC CK2 clock output by 4. If the HPC is using a 16 MHz crystal, this results in MCKL-x being 2 MHz.

BCLK-x is obtained by dividing CK2 by 64. This gives an effective value for BCLK-x of 125 kHz. Note that MCLK-x is inverted before being fed to the codec. This is done to synchronize MCLK-x and BCLK-x on their leading edges.

FS-x and FS-r are the same clocks in the circuit. They are obtained by dividing BCLK-x by 16 using the timer T2 on the HPC. BCLK-x is used as the external clock input on pin T2IO of the HPC and FS-x (FS-r) is obtained from the timer synchronous output TS0. Note that the delay inherent in the HPC between the underflow of a timer and the toggling of the corresponding output allows FS-x and BCLK-x to be leading edge synchronized (more accurately, the delay is within the codec's acceptable limits.) Note that in order to accomplish these functions, the HPC pins need to be properly configured. This is not described here. Please refer to the appropriate HPC documentation and consult the sample program included with this report.

2. **MICROWIRE/PLUS Interface Considerations.** MICROWIRE/PLUS is a National Semiconductor defined 8-bit serial synchronous communication interface. It is designed to allow easy interfacing of NSC microcontrollers and peripheral chips. The HPC microcontroller has a MICROWIRE/PLUS interface; however the TP3054 codec does not. Thus some external "glue logic" is necessary to allow the HPC and the TP3054 to be interfaced.

The HPC MICROWIRE/PLUS interface is operated in Slave mode for this application. This means that the shift clock needed to latch-in/shift-out data from the Micro-wire SIO register is provided externally on the SK pin. Micro-wire latches in data on the leading edge of the SK clock and shifts out data on the trailing edge of SK. Also SK needs to be a burst clock for proper operation.



TL/DD/9246–2

FIGURE 2. Timing Waveforms

The codec shifts out data on the D-x pin on the first 8 leading edges of BCLK-x after a FS-x leading edge. Also, it latches in data on the D-r pin on the first 8 trailing edges of BCLK-x after a FS-r leading edge. Note that FS-x and FS-r are the same in this application. Refer to the timing diagram in *Figure 2*.

Thus, it is seen that there is a timing difference in the way the codec and the Micro-wire interfaces work. However, as seen in *Figure 2*, if the shift clock, SK, to the Microwire interface is delayed with respect to BCLK-x, the two interfaces should work compatibly. This delay is accomplished by clocking BCLK-x through a shift register using MCLK-x as the clock source. This can be seen in the circuit schematic in *Figure 1*. (The author thanks Mr. Richard Lazovick for this suggestion.)

μ-LAW TO LINEAR/LINEAR TO μ-LAW CONVERSION

It was explained earlier that the codec outputs digital values that are companded using the MU-255 PCM standard. However, for linear digital filtering applications, the input needs to be in linear PCM format. Similarly, it is necessary to provide the conversion from linear PCM to MU-255 PCM before output to the codec. The HPC accomplishes this in software.

1. μ-law to linear conversion. The codec output is actually the complement of the μ-law value. Thus, this first needs to be complemented to obtain the true μ-law value. The simplest way to obtain the corresponding linear value is through table look-up. The output of the table is the 16-bit 2's complement linear value. The sample program included with this report utilizes this technique. A macro that constructs this table is also provided.
2. Linear to μ-law conversion. An algorithm to convert a 13-bit positive linear number to 7-bit μ-law is described in *Figure 3*. The algorithm is based on the description in the book by Bellamy listed in the reference. The most significant 8th bit for the μ-law code is obtained from the sign of the input linear code.

1. Get 13-bit positive input value.
2. Add to it the bias value of 31-decimal.
3. The compressed μ-law word is then obtained as follows:

Biased Linear Value Bits												
12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	Q3	Q2	Q1	Q0	a
0	0	0	0	0	0	1	Q3	Q2	Q1	Q0	a	b
0	0	0	0	0	1	Q3	Q2	Q1	Q0	a	b	c
0	0	0	0	1	Q3	Q2	Q1	Q0	a	b	c	d
0	0	0	1	Q3	Q2	Q1	Q0	a	b	c	d	e
0	0	1	Q3	Q2	Q1	Q0	a	b	c	d	e	f
0	1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g
1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g	h

μ-Law Value Bits						
6	5	4	3	2	1	0
0	0	0	Q3	Q2	Q1	Q0
0	0	1	Q3	Q2	Q1	Q0
0	1	0	Q3	Q2	Q1	Q0
0	1	1	Q3	Q2	Q1	Q0
1	0	0	Q3	Q2	Q1	Q0
1	0	1	Q3	Q2	Q1	Q0
1	1	0	Q3	Q2	Q1	Q0
1	1	1	Q3	Q2	Q1	Q0

FIGURE 3. 13-Bit Linear to 8-Bit μ-Law Conversion

POSSIBLE APPLICATIONS

The codec/HPC interface described above can be used in a number of speech processing applications. One application, ADPCM coding of speech, is presently under development. Other applications include: a voiced/unvoiced/silence classifier, a voice pitch tracker, speech detection circuitry etc. Note that the main limitation here (at least for real-time applications) is the amount of effective computation that can be done by the HPC between samples.

REFERENCES

1. National Semiconductor Corporation, *Telecommunications Databook,* Santa Clara, California, 1984.
2. National Semiconductor Corporation, *HPC Programmers Reference Manual,* Santa Clara, California, 1986.
3. National Semiconductor Corporation, *HPC Hardware Reference Manual,* Santa Clara, California, 1986.
4. J. C. Bellamy, *Digital Telephony,* John Wiley & Sons, New York, 1982.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communication package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162

Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

PROGRAM TO TEST CODEC INTERFACE

```
NATIONAL SEMICONDUCTOR CORPORATION   Page:  1
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
TSTCDC

  1                ;
  2                ;
  3                                .TITLE TSTCDC
  4                ;
  5 01C0                           YOFK = M(01C0)          ; OUTPUT SAMPLE STORAGE.
  6 00C0                           PSW = M(00C0)
  7 00D0                           ENIR = M(00D0)
  8 00D2                           IRPD = M(00D2)
  9 00D4                           IRCD = M(00D4)
 10 00D6                           SIO = M(00D6)
 11 00D8                           PORTI = M(00D8)
 12 00E2                           PORTBL = M(00E2)
 13 00E3                           PORTBH = M(00E3)
 14 00E2                           PORTB = W(00E2)
 15 00F2                           DIRBL = M(00F2)
 16 00F3                           DIRBH = M(00F3)
 17 00F2                           DIRB = W(00F2)
 18 00F4                           BFUNL = M(00F4)
 19 00F5                           BFUNH = M(00F5)
 20 00F4                           BFUN = W(00F4)
 21 0188                           T2TIM = W(0188)
 22 0186                           T2REG = W(0186)
 23 018E                           DIVBYL = M(018E)
 24 018F                           DIVBYH = M(018F)
 25 018E                           DIVBY = W(018E)
 26 0190                           TMMDL = M(0190)
 27 0191                           TMMDH = M(0191)
 28 0190                           TMMD = W(0190)
 29               ;
 30               ;
 31               ;
 32                                .MACRO MUTBL,STADR
 33               ;
 34               ; MACRO TO CREATE LOOKUP TABLE FOR MU-255 LAW PCM TO LINEAR CONVERSION.
 35               ; STADR IS THE STARTING ADDRESS FOR THE TABLE, AND MUST BE AN EVEN ADDRESS.
 36               ; THE TABLE OCCUPIES 512 BYTES.
 37               ;
 38                                . = STADR
 39                                .SET SVAL,021
 40                                .SET INCRM,02
 41                                .DO 08
 42                                      .SET MVAL,SVAL-021
 43                                      .DO 010
 44                                        .WORD MVAL
 45                                        .SET MVAL,MVAL+INCRM
 46                                      .ENDDO
 47                                      .SET SVAL,SVAL*02
 48                                      .SET INCRM,INCRM*02
 49                                .ENDDO
 50               ;
 51                                .SET SVAL, 021
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  2
HPC CROSS ASSEMBLER, REV:C,30 JUL 86
TSTCDC

 52                              .SET INCRM, 02
 53                              .DO 08
 54                                    .SET MVAL,SVAL-021
 55                                    .DO 010
 56                                      .SET RVAL,-1*MVAL
 57                                      .WORD RVAL
 58                                      .SET MVAL,MVAL+INCRM
 59                                    .ENDDO
 60                                    .SET SVAL,SVAL*02
 61                                    .SET INCRM,INCRM*02
 62                              .ENDDO
 63             ;
 64                              .ENDM
 65             ;
 66             ;
 67             ;
 68                              .LOCAL
 69 F000                        MUTBL, 0F000
 70             ;
 71 F200                        .= 0F200
 72             CODEC:
 73 F200 B701F0C4                LD SP, 01F0             ; INITIALIZE STACK POINTER.
 74             ;
 75 F204 3059                    JSR INITCD              ; INITIALIZE THE CODEC
 76             FLOOP:
 77 F206 3005                    JSR INPUT               ; GET INPUT SAMPLE, OUTPUT
 78                                                      ; PREVIOUS SAMPLE.
 79 F208 E7                      SHL A
 80 F209 E7                      SHL A
 81 F20A 301F                    JSR OUTPUT              ; CONVERT OUTPUT VALUE TO
 82                                                      ; MU-255 LAW AND SAVE.
 83 F20C 66                      JP FLOOP                ; GO DO NEXT SAMPLE.
 84             ;
 85             ;
 86             INPUT:
 87 F20D B601C088                LD A, YOFK              ; GET DATA TO BE OUTPUT.
 88             NOTDN:
 89 F211 96D210                  IF IRPD,0               ; IS MICROWIRE DONE?
 90 F214 41                      JP MWDONE               ; YES, SO GET DATA.
 91 F215 64                      JP NOTDN                ; NO, SO TRY AGAIN.
 92             MWDONE:
 93 F216 BED6                    X A, SIO                ; GET NEW SAMPLE, OUTPUT
 94                                                      ; COMPUTED DATA.
 95 F218 01                      COMP A                  ; TAKE CARE OF CODEC INVERSION.
 96 F219 99FF                    AND A, 0FF
 97 F21B E7                      SHL A
 98 F21C BAF000                  OR A,0F000              ; FORM MU-LAW TO LINEAR
 99                                                      ; TABLE ADDRESS.
100 F21F AECE                    X A, X
101 F221 D0                      LD A, M(X+)             ; GET LINEAR VALUE
102 F222 AECA                    X A, K
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC

103 F224 04                    LD A, M(X)            ; A BYTE AT A TIME.
104 F225 BCC8CB                LD H(K), L(A)
105 F228 ABCA                  LD A, K
106 F22A 3C                    RET
107              ;
108              ;
109              OUTPUT:
110 F22B 96D41F                RESET IRCD.7
111 F22E E7                    SHL A                 ; SIGN BIT TO C.
112 F22F 06                    IFN C                 ; IS IT POSITIVE?
113 F230 45                    JP OPOS
114 F231 96D40F                SET IRCD.7
115 F234 01                    COMP A
116 F235 04                    INC A                 ; NEGATIVE, SO TAKE 2'S
117                                                  ; COMPLEMENT.
118              OPOS:
119 F236 B80108                ADD A, 0108           ; ADD BIAS.
120 F239 9107                  LD K, 07              ; SET UP COUNTER.
121              ALIGN:                              ; LOOP AND LOCATE MS 1 BIT.
122 F238 E7                    SHL A
123 F23C 07                    IF C
124 F23D 44                    JP ODONE              ; FOUND MS 1 BIT.
125 F23E AACA                  DECSZ K
126 F240 65                    JP ALIGN
127 F241 E7                    SHL A                 ; HAS TO BE 1 IN C NOW.
128              ODONE:
129 F242 AECA                  X A, K
130 F244 E7                    SHL A
131 F245 E7                    SHL A
132 F246 E7                    SHL A
133 F247 E7                    SHL A                 ; COUNTER VALUE IN BITS 4-6.
134 F248 AECC                  X A, B
135 F24A 00                    CLR A
136 F24B 88CB                  LD A, H(K)
137 F24D 3B                    SWAP A
138 F24E 990F                  AND A , 0F
139 F250 96CCFA                OR A, B
140 F253 96D417                IF IRCD.7
141 F256 96C80F                SET A.7
142 F259 01                    COMP A
143 F25A B601C08B              ST A, YOFK
144 F25E 3C                    RET
145              ;
146              INITCD:
147 F25F B7FFB7F2              LD DIRB, 0FFB7        ; SET B3 (T2IO) AND B6 (SK)
148                                                  ; ON PORT B AS INPUTS. SET ALL
149                                                  ; OTHER PINS ON B AS OUTPUT.
150 F263 B70000E2              LD PORTB, 0           ; OUTPUT 0 ON ALL PORT B PINS.
151 F267 96F40B                SET BFUNL.3           ; ALT. FUN. ON B3-T2IO.
152 F26A 96F40D                SET BFUNL.5           ; ALT. FUN. ON B5-SO.
153 F26D 96F508                SET BFUNH.0           ; ALT. FUN. ON B8-TSO.
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC

154 F270 9700D0                 LD ENIR, 0            ; DISABLE INTRPTS.
155 F273 9700D4                 LD IRCD,0             ; SELECT SLAVE MODE FOR M-WIRE.
156 F276 83070188AB             LD T2TIM, 07          ; LOAD 7-DEC INTO T2 TIMER.
157 F27B 83070186AB             LD T2REG, 07          ; LOAD 7-DEC INTO T2 REG.
158 F280 8300018F8B             LD DIVBYH, 0          ; SELECT EXT, CLOCK FOR T2 TIMER.
159             ;
160 F285 8ED6                   X A, SIO
161 F287 8740400190AB           LD TMMD,04040         ; START TIMER T2.
162 F28D 3C                     RET
163             ;
164             ;
165 FFFE 00F2                   .END CODEC
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
SYMBOL TABLE

A       00C8 W        ALIGN   F23B          B       00CC W        BFUN    00F4 W*
BFUNH   00F5 M        BFUNL   00F4 M        CODEC   F200          DIRB    00F2 W
DIRBH   00F3 M*       DIRBL   00F2 M*       DIVBY   018E W*       DIVBYH  018F M
DIVBYL  018E M*       ENIR    00D0 M        FLOOP   F206          INCRM   0200
INITCD  F25F          INPUT   F20D          IRCD    00D4 M        IRPD    00D2 M
K       00CA W        MVAL    205F          MWDONE  F216          NOTDN   F211
ODONE   F242          OPOS    F236          OUTPUT  F22B          PC      00C6 W
PORTB   00E2 W        PORTBH  00E3 M*       PORTBL  00E2 M*       PORTI   00D8 M*
PSW     00C0 M*       RVAL    EDA1          SIO     00D6 M        SP      00C4 W
SVAL    2100          T2REG   D186 W        T2TIM   0188 W        TMMD    0190 W
TMMDH   0191 M*       TMMDL   0190 M*       X       00CE W        YOFK    01C0 M
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
MACRO TABLE

MUTBL

  NO WARNING LINES

  NO ERROR LINES

  656 ROM BYTES USED

SOURCE CHECKSUM = 81D3
OBJECT CHECKSUM = 0C3C

INPUT   FILE C:CODECTST.MAC
LISTING FILE C:CODECTST.PRN
OBJECT  FILE C:CODECTST.LM
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
SYMBOL TABLE

A       00C8 W        ALIGN   F23B          B       00CC W        BFUN    00F4 W*
BFUNH   00F4 M        BFUNL   00F4 M        CODEC   F200          DIRB    00F2 W
DIRBH   00F3 M*       DIRBL   00F2 M*       DIVBY   018E W*       DIVBYH  018F M
DIVBYL  01B3 M*       ENIR    00D0 M        FLOOP   F206          INCRM   0200
INITCD  F25F          INPUT   F20D          IRCD    00D4 M        IRPD    00D2 M
K       00CA W        MVAL    205F          MWDONE  F216          NOTDN   F211
ODONE   F242          OPOS    F236          OUTPUT  F22B          PC      00C6 W
PORTB   00E2 W        PORTBH  00E3 M*       PORTBL  00E2 M*       PORTI   00D8 M*
PSW     00C0 M*       RVAL    EDA1          SIO     00D6 M        SP      00C4 W
SVAL    2100          T2REG   D186 W        T2TIM   0188 W        TMMD    0190 W
TMMDH   0191 M*       TMMDL   0190 M*       X       00CE W        YOFK    01C0 M
```

Digital Filtering Using the HPC

National Semiconductor
Application Note 485
Ashok Krishnamurthy

INTRODUCTION

This report discusses the implementation of Infinite Impulse Response (IIR) digital filters using the National Semiconductor HPC microcontroller. A general program, that can be used to implement cascaded second order sections, up to a maximum of 8 sections, is also included. The program may have to be modified for specific A/D and D/A interfaces.

This report is not intended to be a tutorial on Digital Filter Design methods or their implementation details. Such information can be found in references 1 and 2 below. The general discussion included here closely follows that in reference 3.

DIGITAL FILTERING

The general IIR filter with input x (n) and output y (n) can be described by a transfer function of the form

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a(0) + a(1)\,z^{-1} + \ldots + a(m)\,z^{-m}}{1 + b(1)\,z^{-1} + \ldots + b(p)\,z^{-p}}$$

To minimize the effects of coefficient truncation, high order filters are usually implemented as a cascade of second order sections. (Another possible choice is parallel realization—see references below).

In cascade realizations, the numerator and denominator polynomials in the above are factored into second order terms, and the filter is realized as a cascade of such second order sections. This is shown in *Figure 1*. A typical second order section has a transfer function of the form

$$H(z) = \frac{A0 + A1 \times z^{-1} + A2 \times z^{-2}}{1 + B1 \times z^{-1} + B2 \times z^{-2}}$$

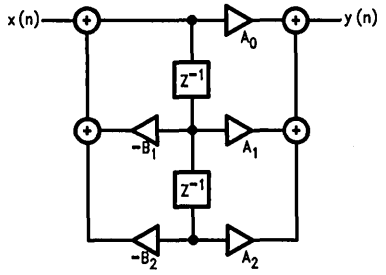
A second order section such as the above can be realized in a number of ways; the one of concern here is the so-called 1-D form (see Reference 3). The second order 1-D form is shown in *Figure 2*. Based on this figure, we can obtain the following equations:

$$m(k) = x(k) - B1 \times m(k-1) - B2 \times m(k-2)$$

$$y(k) = A0 \times m(k) + A1 \times m(k-1) + A2 \times m(k-2)$$

Define $T1 = -B1 \times m(k-1) - B2 \times m(k-2)$,

$T2 = A1 \times m(k-1) + A2 \times m(k-2)$



TL/DD/9247-2

FIGURE 2. One Second Order Section

Since T1 and T2 depend on signal values at time k − 1 and k − 2, we can precompute and store these quantities in the time interval from k − 1 to k. Then, when x(k) becomes available at time k, y(k) and m(k) can be quickly computed using

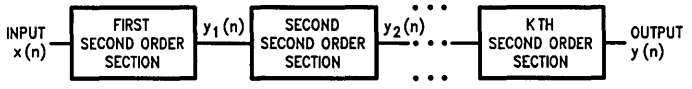
$$m(k) = x(k) + T1,$$

$$y(k) = A0 \times m(k) + T2$$

If there are a number of stages, then these computations should be repeated for each stage. Based on these discussions, the operation of a digital filter can be described using the flowchart in *Figure 3*.

USING THE FILTER PROGRAM

Appendix A contains the listing of the program FILTER that can be used to implement cascaded IIR filters as described above. The program as shown uses a codec interfaced to the HPC using MICROWIRE/PLUS™ to do the A/D and D/A conversion. The program can be used with other A/D and D/A converters by suitably modifying the following subroutines: INPUT, OUTPUT and INIT. Only the portions of INIT that deal with the codec interface need to be modified.



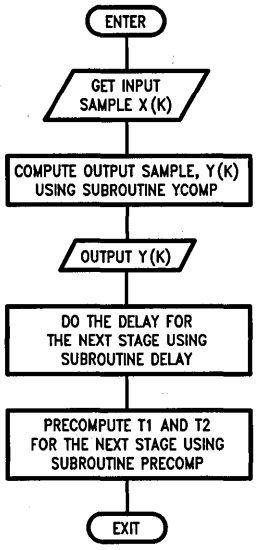
TL/DD/9247-1

FIGURE 1. Cascade Realization of a Digital Filter

The filter coefficients and the number of cascaded stages need to be supplied to the program. This is done as follows:

1. *Specification of filter order.* Define a word address called ROMNST and store the number of cascaded stages in that word. The program is presently set up for 4 cascaded stages.
2. *Specification of filter coefficients.* Each second order stage needs the specification of 5 coefficients, A0, A1, A2, B1 and B2. If the number of stages is m, let the coefficients be

   A0–1, A1–1, A2–1, B1–1, B2–1 for stage 1,
   A0–2, A1–2, A2–2, B1–2, B2–2 for stage 2,
   .
   .
   .
   A0–m, A1–m, A2–m, B1–m, B2–m for stage m.



TL/DD/9247–3

FIGURE 3. Flowchart for the Computations in a Second Order Module (Based on Reference 3)

Define 5 word addresses called ROMA0, ROMA1, ROMA2, ROMB1, ROMB2 and store these coefficients at these addresses as follows:

ROMA0: .WORD A0–1, A0–2, A0–3, ... A0–m
ROMA1: .WORD A1–1, A1–2, A1–3, ... A1–m
ROMA2: .WORD A2–1, A2–2, A2–3, ... A2–m
ROMB1: .WORD B1–1, B1–2, B1–3, ... B1–m
ROMB2: .WORD B2–1, B2–2, B2–3, ... B2–m.

Note that the coefficients are signed and need to be in 2's complement representation. Also, the stored coefficients need to be half their actual value. This is because of the way that the program does 2's complement multiplication using the subroutine SMULT.

The FILTER program copies all the coefficients to on-chip RAM for faster execution. Also temporary storage for m (k), m (k − 1), m (k − 2), T1 and T2 is obtained from on-chip RAM. This, along with the storage of various addresses used by the program consumes the entire 192 bytes of user base page RAM.

Note that the filter program does not check for overflow during the various additions. This is because the HPC does not have a signed addition/subtraction overflow flag, and it was felt that the simulation of this feature in software would add excessive overhead. It is therefore the user's responsibility to ensure that the filter coefficients are properly scaled so that the overflow will not occur.

16 x 16 2's COMPLEMENT MULTIPLICATION

One of the basic operations in digital filtering is that of signed multiplication. Since the HPC supports unsigned multiplication only, a method to perform 2's complement multiply using the unsigned multiply is needed.

Let A and B be 2's complement 16 bit integers. Consider the following cases.

1. $A \geq 0$, $B \geq 0$. In this case the unsigned multiply result is A×B, which is also the 2's complement multiply result. Thus no further processing is needed.
2. $A \geq 0$, $B < 0$. In this case the unsigned multiply result is $(2^{16}) \times A - A \times |B|$. However the desired result is $(2^{32}) - A \times |B|$. Thus we need to add $(2^{32}) - (2^{16}) \times A$ to the unsigned multiply result to obtain the correct value.
3. $A < 0$, $B \geq 0$. This case is similar to the previous one. $(2^{32}) - (2^{16}) \times B$ should be added to the unsigned multiply result to get the correct answer.
4. $A < 0$, $B < 0$. The unsigned multiply result in this case is $(2^{32}) - (2^{16}) \times (|A| + |B|) + |A| \times |B|$. The desired result in this case is $|A| \times |B|$. To get the correct answer, add $(2^{16}) \times (|A| + |B|)$ to the unsigned multiply result.

Based on the above discussion, an algorithm for 2's complement multiplication, where the result is a 32 bit 2's complement integer is shown in *Figure 4.*

1. Let A and B be the two 2's complement integers to be multiplied.
2. Compute C = A × B, the unsigned product of A and B. Let the upper half of C be C-hi and its lower half C-lo.
3. If A is negative, then add (2^{16}) − B to C-hi. This can be easily done using the SET C, SUBC instructions of the HPC. Let the result be C-hi1.
4. If B is negative, then add (2^{16}) − A to C-hi1. Again it is easily done using the SET C, SUBC instructions. Let the result be C-hi2.
5. The 2's complement product of A and B is C-hi2. C-lo.

FIGURE 4. Algorithm for 2's Complement Multiplication.

MULTIPLICATION BY FILTER COEFFICIENTS

The coefficients that arise in most IIR filter designs are numbers that are usually in the range from $-2 < \text{coeff} < 2$. The coefficients, in most instances can be scaled to be in this range. The action of digital filtering involves successive multiplications. If we want no loss in accuracy due to multiplication, the word length needed to store successive partial products increases rapidly—clearly an impractical choice. Thus the results of the multiplication at the various stages need to be truncated to 16 bits before proceeding to the next stage. The program FILTER does this as follows: The filter state variables are regarded as integers, while the filter coefficients are regarded as fixed point fractions with the binary point to the immediate right of the sign bit. After the multiplication, the result is shifted so that the integer part of the product is in one word, and the fractional part in another. The integer part is then returned as the result of the multiplication, i.e. the product is truncated to 16 bits. This is performed by the subroutine SMULT. Since the filter coefficients are regarded as fixed point fractions, only coefficients in the range $-1 < \text{coeff} < 1$ can be represented. However, as discussed earlier, the coefficients are usually in the $-2 < \text{coeff} < 2$ range. This is handled by storing half the coefficient value, and SMULT performs a multiplication by 2 (Shift left) to compensate for it. This is why the coefficient values need to be half their value—a fact mentioned earlier.

REFERENCES

1. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
2. L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
3. H.T. Nagle and V.P. Nelson, "Digital Filter Implementation on 16-bit Microcomputers", *IEEE Micro,* Feb. 1981, pp. 23–41.

APPENDIX A

Listing of Code for the Program FILTER

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  1
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

 1                ;
 2                ; THIS IS A DEMO PROGRAM TO ILLUSTRATE THE IMPLEMENTATION OF A DIGITAL
 3                ; FILTER ON THE HPC. THE PROGRAM CAN BE USED TO IMPLEMENT CASCADED
 4                ; SECOND ORDER STAGES. THE MAXIMUM NUMBER OF CASCADED STAGES POSSIBLE
 5                ; IS 8 (I.E. THE MAXIMUM FILTER ORDER IS 16).
 6                ;
 7                ; THE PROGRAM IS DESIGNED FOR THE ANALOG INTERFACE BEING THROUGH
 8                ; A CODEC. THE CODEC OUTPUT AND INPUT ARE INTERFACED TO THE HPC USING
 9                ; MICROWIRE/PLUS. THIS RESTRICTS THE SAMPLING RATE TO 8 KHZ. ALSO, AT
10                ; THIS SAMPLING RATE, THE HPC CAN ONLY IMPLEMENT A SECOND ORDER FILTER.
11                ; IF A DIFFERENT ANALOG INTERFACE THAT ALLOWS A LOWER SAMPLING RATE IS
12                ; USED, HIGHER ORDER FILTERS CAN BE IMPLEMENTED. THIS WILL INVOLVE CHANGES
13                ; TO THE FOLLOWING SUBROUTINES: INPUT, OUTPUT AND THE PORTIONS OF INIT
14                ; CONCERNED WITH CODEC INITIALIZATION.
15                ;
16                ; THE PROGRAM IS BASED ON THE DESCRIPTION GIVE IN:
17                ;
18                ;       H.T. NAGLE AND V.P. NELSON, "DIGITAL FILTER IMPLEMENTATION
19                ;       ON 16-BIT MICROCOMPUTERS," IEEE MICRO, FEB. 1981, 23-41.
20                ;
21                ;
22                              .TITLE FILTER
23                ;
24                ; DEFINE FILTER VARIABLES AND STORAGE.
25                ;
26      0000                    YOUT = M(00)            ; OUTPUT SAMPLE STORAGE.
27      0002                    YOFK = W(02)            ; TEMPORARY STORAGE.
28      0004                    NSTG = W(04)            ; NUMBER OF FILTER STAGES.
29      0006                    NCNT = W(06)            ; TEMPORARY STORAGE.
30      0008                    PTEMP = W(08)           ; TEMPORARY STORAGE.
31      000A                    MTEMP = W(0A)           ; TEMPORARY STORAGE.
32      000C                    A0ADDR = W(0C)          ; ADDRESS OF START OF A0 AREA.
33      000E                    A1ADDR = W(0E)          ; ADDR. OF START OF A1 AREA.
34      0010                    A2ADDR = W(010)         ; ADDR. OF START OF A2 AREA.
35      0012                    B1ADDR = W(012)         ; ADDR. OF START OF B1 AREA.
36      0014                    B2ADDR = W(014)         ; ADDR. OF START OF B2 AREA.
37      0016                    M0ADDR = W(016)         ; ADDR. OF START OF M0 AREA.
38      0018                    M1ADDR = W(018)         ; ADDR. OF START OF M1 AREA.
39      001A                    M2ADDR = W(01A)         ; ADDR. OF START OF M2 AREA.
40      001C                    T1ADDR = W(01C)         ; ADDR. OF START OF T1 AREA.
41      001E                    T2ADDR = W(01E)         ; ADDR. OF START OF T2 AREA.
42                ; MAXIMUM NUMBER OF STAGES IS 8.
43      0020                    A0 = W(020)             ; COEFF. A0.
44      0030                    A1 = W(030)             ; COEFF. A1.
45      0040                    A2 = W(040)             ; COEFF. A2.
46      0050                    B1 = W(050)             ; COEFF. B1.
47      0060                    B2 = W(060)             ; COEFF. B2.
48      0070                    M0 = W(070)             ; M(K).
49      0080                    M1 = W(080)             ; M(K-1).
50      0090                    M2 = W(090)             ; M(K-2).
51      00A0                    T1 = W(0A0)             ; T1.
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  2
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

52       00B0                        T2 = W(0B0)              ; T2.
53                 ;
54                 ; DEFINITION OF HPC REGISTER NAMES.
55                 ;
56       00C0                        PSW = M(00C0)
57       00D0                        ENIR = M(00D0)
58       00D2                        IRPD = M(00D2)
59       00D4                        IRCD = M(00D4)
60       00D6                        SIO = M(00D6)
61       00D8                        PORTI = M(00D8)
62       00E2                        PORTBL = M(00E2)
63       00E3                        PORTBH = M(00E3)
64       00E2                        PORTB = W(00E2)
65       00F2                        DIRBL = M(00F2)
66       00F3                        DIRBH = M(00F3)
67       00F2                        DIRB = W(00F2)
68       00F4                        BFUNL = M(00F4)
69       00F5                        BFUNH = M(00F5)
70       00F4                        BFUN = W(00F4)
71       0188                        T2TIM = W(0188)
72       0186                        T2REG = W(0186)
73       018E                        DIVBYL = M(018E)
74       018F                        DIVBYH = M(018F)
75       018E                        DIVBY = W(018E)
76       0190                        TMMDL = M(0190)
77       0191                        TMMDH = M(0191)
78       0190                        TMMD = W(0190)
79                 ;
80                 ; INCLUDE THE MU-LAW TO LINEAR CODE CONVERSION TABLE.
81                 ;
82                                   .INCLD MUTBL.MAC
83 F000                              MUTBL, 0F000
84                 ;
85  F200                            . = 0F200
86                 FILTER:
87 F200 B701F0C4                    LD SP, 01F0              ; INITIALIZE STACK POINTER.
88 F204 305A                        JSR INIT                 ; INITIALIZE THE CODEC
89                                                           ; AND FILTER VARIABLES.
90                 ; NEXT COMES THE BASIC FILTER LOOP.
91                 FLOOP:
92 F206 3101                        JSR INPUT                ; GET INPUT SAMPLE, OUTPUT
93                                                           ; PREVIOUS FILTER OUTPUT.
94 F208 311B                        JSR YCOMP                ; COMPUTE NEW OUTPUT.
95 F20A 315B                        JSR OUTPUT               ; CONVERT OUTPUT VALUE TO
96                                                           ; MU-255 LAW AND SAVE.
97 F20C 31AA                        JSR PRECOM               ; PRECOMPUTE FOR NEXT SAMPLE.
98 F20E 68                          JP FLOOP                 ; GO DO NEXT SAMPLE.
99                                  .LOCAL
100                ;
101                ;
102                ;
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  3
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                        ; THIS SUBROUTINE COMPUTES 2*F*I, WHERE F IS A 2'S COMPLEMENT
                        ; BINARY FRACTION AND I IS A 2'S COMPLEMENT INTEGER. THE INTEGER
                        ; PART OF THE PRODUCT IS RETURNED IN A. ON INPUT, EITHER F OR I
                        ; SHOULD BE IN A AND THE ADDRESS OF THE OTHER IN B.
                        ;
                        ;
                        SMULT:
F20F B700000A                       LD MTEMP, 0             ; CLEAR TEMPORARY STORAGE.
F213 A9CC                           INC B                   ; B NOW POINTS TO UPPER BYTE
                                                            ; OF MULTIPLIER.
F215 17                             IF M(B).7               ; IS IT NEGATIVE?
F216 AB0A                           ST A, MTEMP             ; THEN SAVE MULTIPLICAND IN MTEMP.
F218 AACC                           DECSZ B                 ; B INTO WORD POINTER.
F21A 40                             NOP
F21B AE0A                           X A, MTEMP              ; SWAP A AND MTEMP.
F21D 960B17                         IF M(($MTEMP) + 1).7    ; IS MULTIPLICAND NEGATIVE?
F220 F8                             ADD A, W(B)             ; THEN ACCUMULATE MULTIPLIER
F221 AE0A                           X A, MTEMP
F223 FE                             MULT A, W(B)            ; UNSIGNED MULTIPLY.
F224 AECE                           X A, X                  ; UPPER HALF IN A.
F226 02                             SET C
F227 960AEB                         SUBC A, MTEMP
F22A E7                             SHL A
F22B 96CF17                         IF H(X).7
F22E 04                             INC A
F22F E7                             SHL A
F230 96CF16                         IF H(X).6
F233 04                             INC A
F234 3C                             RET
                        ;
                        ;
                        ; THIS SUBROUTINE PERFORMS THE INITIALIZATION FOR THE FILTER.
                        ; IT DOES THE FOLLOWING:
                        ;               1. SET UP THE FILTER VARIABLES.
                        ;               2. COPY THE FILTER COEFFS. FROM ROM TO ON CHIP RAM.
                        ;               3. INITIALIZE AND START THE CODEC.
                        ;
                        ;
                        ; DEFINE FILTER COEFFICIENTS.
                        ;
F235 40                             .EVEN
F236 0400       ROMNST:             .WORD 4
F238 C430       ROMA0:              .WORD 12484, 3217, 4574, 7636
    F23A 910C
    F23C DE11
    F23E D41D
F240 C430       ROMA1:              .WORD 12484, 3217, 4574, 7636
    F242 910C
    F244 DE11
    F246 D41D
F248 C430       ROMA2:              .WORD 12484, 3217, 4574, 7636
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  4
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

    F24A 910C
    F24C DE11
    F24E D41D
148 F250 B939       ROMB1:          .WORD 14777, 9826, 19308, 11207
    F252 6226
    F254 6C4B
    F256 C72B
149 F258 A1D5       ROMB2:          .WORD -10847, -15783, -6940, -14068
    F25A 59C2
    F25C E4E4
    F25E 0CC9
150                 INIT:
151 F260 B6F236A8                   LD A, W(ROMNST)
152 F264 AB04                       ST A, NSTG              ; SET UP NO. OF STAGES.
153 F266 9020                       LD A, $A0
154 F268 AB0C                       ST A, A0ADDR            ; COPY ADDRESS OF A0 AREA.
155 F26A 9030                       LD A, $A1
156 F26C AB0E                       ST A, A1ADDR            ; COPY ADDRESS OF A1 AREA.
157 F26E 9040                       LD A, $A2
158 F270 AB10                       ST A, A2ADDR            ; COPY ADDRESS OF A2 AREA.
159 F272 9050                       LD A, $B1
160 F274 AB12                       ST A, B1ADDR            ; COPY ADDRESS OF B1 AREA.
161 F276 9060                       LD A, $B2
162 F278 AB14                       ST A, B2ADDR            ; COPY ADDRESS OF B2 AREA.
163 F27A 9070                       LD A, $M0
164 F27C AB16                       ST A, M0ADDR            ; COPY ADDRESS OF M0 AREA.
165 F27E 9080                       LD A, $M1
166 F280 AB18                       ST A, M1ADDR            ; COPY ADDRESS OF M1 AREA.
167 F282 9090                       LD A, $M2
168 F284 AB1A                       ST A, M2ADDR            ; COPY ADDRESS OF M2 AREA.
169 F286 90A0                       LD A, $T1
170 F288 AB1C                       ST A, T1ADDR            ; COPY ADDRESS OF T1 AREA.
171 F28A 9080                       LD A, $T2
172 F28C AB1E                       ST A, T2ADDR            ; COPY ADDRESS OF T2 AREA.
173                 ;
174                 ; COPY THE A0 COEFFS. TO ON-CHIP RAM.
175                 ;
176 F28E B3F238                     LD X, ROMA0
177 F291 9220                       LD B, $A0
178 F293 AC04CA                     LD K, NSTG
179                 CAOLP:
180 F296 F0                         LD A, W(X+)
181 F297 E1                         XS A, W(B+)
182 F298 40                         NOP
183 F299 AACA                       DECSZ K
184 F29B 65                         JP CAOLP
185                 ;
186                 ; COPY THE A1 COEFFS. TO ON-CHIP RAM.
187 F29C B3F240                     LD X, ROMA1
188 F29F 9230                       LD B, $A1
189 F2A1 AC04CA                     LD K, NSTG
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  5
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

190                 CA1LP:
191 F2A4 F0                         LD A, W(X+)
192 F2A5 E1                         XS A, W(B+)
193 F2A6 40                         NOP
194 F2A7 AACA                       DECSZ K
195 F2A9 65                         JP CA1LP
196                 ;
197                 ; COPY THE A2 COEFFS. TO ON-CHIP RAM.
198 F2AA B3F248                     LD X, ROMA2
199 F2AD 9240                       LD B, $A2
200 F2AF AC04CA                     LD K, NSTG
201                 CA2LP:
202 F2B2 F0                         LD A, W(X+)
203 F2B3 E1                         XS A, W(B+)
204 F2B4 40                         NOP
205 F2B5 AACA                       DECSZ K
206 F2B7 65                         JP CA2LP
207                 ;
208                 ; COPY THE B1 COEFFS. TO ON-CHIP RAM.
209 F2BB B3F250                     LD X, ROMB1
210 F2BB 9250                       LD B, $B1
211 F2BD AC04CA                     LD K, NSTG
212                 CB1LP:
213 F2C0 F0                         LD A, W(X+)
214 F2C1 E1                         XS A, W(B+)
215 F2C2 40                         NOP
216 F2C3 AACA                       DECSZ K
217 F2C5 65                         JP CB1LP
218                 ;
219                 ; COPY THE B2 COEFFS. TO ON-CHIP RAM.
220 F2C6 B3F258                     LD X, ROMB2
221 F2C9 9260                       LD B, $B2
222 F2CB AC04CA                     LD K, NSTG
223                 CB2LP:
224 F2CE F0                         LD A, W(X+)
225 F2CF E1                         XS A, W(B+)
226 F2D0 40                         NOP
227 F2D1 AACA                       DECSZ K
228 F2D3 65                         JP CB2LP
229                 ;
230                 ; ZERO OUT THE REST OF USER BASE PAGE RAM.
231                 ;
232 F2D4 8D70BE                     LD BK, $MO, 0BE
233                 ZEROLP:
234 F2D7 00                         CLR A
235 F2D8 E1                         XS A, W(B+)
236 F2D9 62                         JP ZEROLP
237                 ;
238                 ;
239                 ; NOW INITIALIZE AND START THE CODEC.
240                 ;
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  6
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

241                 ;
242 F2DA B7FFB7F2                   LD DIRB,0FFB7           ; SET B3 (T2IO) AND B6 (SK)
243                                                         ; ON PORT B AS INPUTS. SET ALL
244                                                         ; OTHER PINS ON B AS OUTPUT.
245 F2DE B70000E2                   LD PORTB, 0             ; OUTPUT 0 ON ALL PORT B PINS.
246 F2E2 96F40B                    SET BFUNL.3             ; ALT. FUN. ON B3-T2IO.
247 F2E5 96F40D                    SET BFUNL.5             ; ALT. FUN. ON B5-SO.
248 F2E8 96F508                    SET BFUNH.0             ; ALT. FUN. ON B8-TSO.
249 F2EB 9700D0                    LD ENIR, 0              ; DISABLE INTRPTS.
250 F2EE 9700D4                    LD IRCD, 0              ; SELECT SLAVE MODE FOR M-WIRE.
251 F2F1 83070188AB                LD T2TIM, 07            ; LOAD 7-DEC INTO T2 TIMER.
252 F2F6 83070186AB                LD T2REG, 07            ; LOAD 7-DEC INTO T2 REG.
253 F2FB 8300018F8B                LD DIVBYH, 0            ; SELECT EXT. CLOCK FOR T2 TIMER.
254                 ;
255 F300 8ED6                      X A, SIO
256 F302 8740400190AB              LD TMMD, 04040          ; START TIMER T2.
257 F308 3C                        RET
258                 ;
259                 ;
260                 ; THIS SUBROUTINE OUTPUTS THE PREVIOUS Y(K) TO THE CODEC AND READS
261                 ; THE NEW INPUT VALUE. THEN THE MU-255 VALUE IS CONVERTED TO LINEAR
262                 ; BY TABLE LOOK UP. THE TABLE IS ASSUMED TO START AT F000.
263                 ;
264                 ;
265                 INPUT:
266 F309 AB02                      LD A, YOFK              ; GET DATA TO BE OUTPUT.
267                 NOTDN:
268 F30B 96D210                    IF IRPD.0               ; IS MICROWIRE DONE?
269 F30E 41                        JP MWDONE               ; YES, SO GET DATA.
270 F30F 64                        JP NOTDN                ; NO, SO TRY AGAIN.
271                 MWDONE:
272 F310 8ED6                      X A, SIO                ; GET NEW SAMPLE, OUTPUT
273                                                         ; COMPUTED DATA.
274 F312 01                        COMP A                  ; TAKE CARE OF CODEC INVERSION.
275 F313 99FF                      AND A, OFF
276 F315 E7                        SHL A
277 F316 BAF000                    OR A, 0F000             ; FORM MU-LAW TO LINEAR
278                                                         ; TABLE ADDRESS.
279 F319 AECE                      X A, X
280 F31B D0                        LD A, M(X+)             ; GET LINEAR VALUE
281 F31C AECA                      X A, K
282 F31E D4                        LD A, M(X)              ; A BYTE AT A TIME.
283 F31F 8CC8CB                    LD H(K), L(A)
284 F322 A8CA                      LD A, K
285 F324 3C                        RET
286                 ;
287                 ;
288                 YCOMP:
289                 ; THIS SUBROUTINE COMPUTES THE OUTPUT SAMPLE Y(K).
290                 ; THE INPUT SAMPLE X(K) IS INPUT IN REG. A.
291                 ; THE OUTPUT IS RETURNED IN REG. A.
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  7
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

292                 ;
293 F325 AC0406                     LD NCNT, NSTG           ; COPY THE NUMBER OF STAGES TO
294                                                         ; NCNT.
295                 YLOOP:
296 F328 AD1CF8                     ADD A, W(T1ADDR)        ; A ≤ X(K) + T1.
297 F32B AD16AB                     ST A, W(MOADDR)         ; M(K) ≤ X(K) + T1.
298 F32E AC0CCC                     LD B, AOADDR            ; B ≤ ADDR(AO).
299 F331 3522                       JSR SMULT               ; A ≤ AO*M(K).
300 F333 AD1EF8                     ADD A, W(T2ADDR)        ; A ≤ AO*M(K) + T2.
301                                                         ;
302 F336 AA06                       DECSZ NCNT              ; DONE ALL STAGES?
303 F338 941B       R               JMP YMORE               ; NO GO DO SOME MORE.
304                                                         ;
305                                                         ; GET HERE MEANS ALL STAGES DONE.
306 F33A AB02                       ST A, YOFK              ; SAVE TEMPORARILY.
307 F33C A804                       LD A, NSTG
308 F33E 05                         DEC A
309 F33F E7                         SHL A
310 F340 01                         COMP A
311 F341 04                         INC A                   ; A ≤ -2*(NSTG-1).
312 F342 A0C81CF8                   ADD T1ADDR, A           ; RESTORE T1ADDR.
313 F346 A0C816F8                   ADD MOADDR, A           ; RESTORE MOADDR.
314 F34A A0C80CF8                   ADD AOADDR, A           ; RESTORE AOADDR.
315 F34E A0C81EF8                   ADD T2ADDR, A           ; RESTORE T2ADDR.
316 F352 A802                       LD A, YOFK              ; A ≤ Y(K).
317 F354 3C                         RET
318                 ;
319                 ; PREPARE FOR NEXT STAGE ITERATION.
320                 ;
321                 YMORE:
322 F355 82021CF8                   ADD T1ADDR, 02
323 F359 820216F8                   ADD MOADDR, 02
324 F35D 82020CF8                   ADD AOADDR, 02
325 F361 82021EF8                   ADD T2ADDR, 02
326 F365 953D                       JMP YLOOP
327                 ;
328                 ; THIS SUBROUTINE CONVERTS THE 16 BIT OUTPUT VALUE TO
329                 ; 8 BIT MU-LAW.
330                 ;
331                 OUTPUT:
332 F367 96D41F                     RESET IRCD.7
333 F36A E7                         SHL A                   ; SIGN BIT TO C.
334 F36B 06                         IFN C                   ; IS IT POSITIVE?
335 F36C 45                         JP OPOS
336 F36D 96D40F                     SET IRCD.7
337 F370 01                         COMP A
338 F371 04                         INC A                   ; NEGATIVE, SO TAKE 2'S
339                                                         ; COMPLEMENT.
340                 ; OPOS:
341 F372 B80108                     ADD A, 0108             ; ADD BIAS.
342 F375 9107                       LD K, 07                ; SET UP COUNTER.
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  8
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

343                 ALIGN:
344 F377 E7                         SHL A                   ; LOOP AND LOCATE MS 1 BIT.
345 F378 07                         IF C
346 F379 44                         JP ODONE                ; FOUND MS 1 BIT.
347 F37A AACA                       DECSZ K
348 F37C 65                         JP ALIGN
349 F37D E7                         SHL A                   ; HAS TO BE 1 IN C NOW.
350                 ODONE:
351 F37E AECA                       X R, K
352 F380 E7                         SHL A
353 F381 E7                         SHL A
354 F382 E7                         SHL A
355 F383 E7                         SHL A                   ; COUNTER VALUE IN BITS 4-6.
356 F384 AECC                       X A, B
357 F386 00                         CLR A
358 F387 88CB                       LD A, H(K)
359 F389 3B                         SWAP A
360 F38A 990F                       AND A, 0F
361 F38C 96CCFA                     OR A, B
362 F38F 96D417                     IF IRCD.7
363 F392 96C80F                     SET A.7
364 F395 01                         COMP A
365 F396 8B00                       ST A, YOUT
366 F398 3C                         RET
367                 ;
368                 ; THIS SUBROUTINE UPDATES M(K-1) AND M(K-2) FOR THE NEXT SAMPLE.
369                 ;
370 F399 AC1ACC                     LD B, M2ADDR            ; B ≤ ADDR(M2),
371 F39C AC04CA                     LD K, NSTG              ; K ≤ NSTG.
372 F39F AC18CE                     LD X, M1ADDR            ; X ≤ ADDR(M1).
373                 DLYLP1:
374 F3A2 F0                         LD A, W(X+)             ; A ≤ M(K-1).
375 F3A3 E1                         XS A, W(B+)             ; M(K-2) ≤ M(K-1).
376 F3A4 40                         NOP
377 F3A5 AACA                       DECSZ K
378 F3A7 65                         JP DLYLP1
379                 ;
380 F3A8 AC18CC                     LD B, M1ADDR            ; B ≤ ADDR(M1),
381 F3AB AC04CA                     LD K, NSTG              ; K ≤ NSTG.
382 F3AE AC16CE                     LD X, M0ADDR            ; X ≤ ADDR(M0).
383                 DLYLP2:
384 F3B1 F0                         LD A, W(X+)             ; A ≤ M(K).
385 F3B2 E1                         XS A, W(B+)             ; M(K-1) ≤ M(K).
386 F3B3 40                         NOP
387 F3B4 AACA                       DECSZ K
388 F3B6 65                         JP DLYLP2
389 F3B7 3C                         RET
390                 ;
391                 ;
392                 PRECOMP:
393                 ; THIS SUBROUTINE PRECOMPUTES T1 AND T2 BEFORE THE NEXT INPUT
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  9
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                    ; SAMPLE ARRIVES.
                    ;
F3B8 AC0406                     LD NCNT, NSTG           ; COPY NO. OF STAGES.
                    PRELP:
F3BB AD18A8                     LD A, W(M1ADDR)         ; A ≤ M(K-1).
F3BE AC12CC                     LD B, B1ADDR            ; B ≤ ADDR(-B1).
F3C1 35B2                       JSR SMULT               ; A ≤ -B1*M(K-1).
F3C3 AB08                       ST A, PTEMP
F3C5 AD1AA8                     LD A, W(M2ADDR)         ; A ≤ M(K-2).
F3C8 AC14CC                     LD B,B2ADDR             ; B ≤ ADDR(-B2).
F3CB 35BC                       JSR SMULT               ; A ≤ -B2*M(K-2).
F3CD 9608F8                     ADD A, PTEMP            ; A ≤ -B1*M(K-1) - B2*M(K-2).
F3D0 AD1CAB                     ST A,W(T1ADDR)
F3D3 AD18A8                     LD A, W(M1ADDR)         ; A ≤ M(K-1).
F3D6 AC0ECC                     LD B, A1ADDR            ; B ≤ ADDR(A1).
F3D9 35CA                       JSR SMULT               ; A ≤ A1*M(K-1).
F3DB AB08                       ST A, PTEMP
F3DD AD1AA8                     LD A, W(M2ADDR)         ; A ≤ M(K-2).
F3E0 AC10CC                     LD B, A2ADDR            ; B ≤ ADDR(A2).
F3E3 3504                       JSR SMULT               ; A ≤ A2*M(K-2).
F3E5 9608F8                     ADD A, PTEMP            ; A ≤ A1*M(K-1) + A2*M(K-2).
                                                        ;
F3E8 AA06                       DECSZ NCNT              ; DONE ALL STAGES?
F3EA 9427                       JMP PMORE               ; NO, GO DO SOME MORE.
                                                        ;
                                                        ; GET HERE MEANS DONE ALL STAGES.
F3EC A804                       LD A, NSTG
F3EE 05                         DEC A
F3EF E7                         SHL A
F3F0 01                         COMP A
F3F1 04                         INC A                   ; A ≤ -2*(NSTG - 1).
F3F2 A0C818F8                   ADD M1ADDR, A           ; RESTORE M1ADDR.
F3F6 A0C81AF8                   ADD M2ADDR, A           ; RESTORE M2ADDR.
F3FA A0C81CF8                   ADD T1ADDR, A           ; RESTORE T1ADDR.
F3FE A0C81EF8                   ADD T2ADDR, A           ; RESTORE T2ADDR.
F402 A0C812F8                   ADD B1ADDR, A           ; RESTORE B1ADDR.
F406 A0C814F8                   ADD B2ADDR, A           ; RESTORE B2ADDR.
F40A A0C80EF8                   ADD A1ADDR, A           ; RESTORE A1ADDR.
F40E A0C810F8                   ADD A2ADDR, A           ; RESTORE A2ADDR.
F412 3C                         RET
                    ;
                    ; PREPARE FOR NEXT STAGE ITERATION.
                    ;
                    PMORE:
F413 820218F8                   ADD M1ADDR, 02          ; UPDATE M1ADDR.
F417 82021AF8                   ADD M2ADDR, 02          ; UPDATE M2ADDR.
F41B 82021CF8                   ADD T1ADDR, 02          ; UPDATE T1ADDR.
F41F 82021EF8                   ADD T2ADDR, 02          ; UPDATE T2ADDR.
F423 820212F8                   ADD B1ADDR, 02          ; UPDATE B1ADDR.
F427 820214F8                   ADD B2ADDR, 02          ; UPDATE B2ADDR.
F42B 82020EF8                   ADD A1ADDR, 02          ; UPDATE A1ADDR.
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  10
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

445 F42F 820210F8                    ADD A2ADDR, 02          ; UPDATE A2ADDR.
446 F433 9578                        JMP PRELP
447                 ;
448                 ;
449 FFFE 00F2                        .END FILTER
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  11
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER
SYMBOL TABLE

A       00C8 W        A0      0020 W        A0ADDR 000C W        A1      0030 W
A1ADDR  000E W        A2      0040 W        A2ADDR 0010 W        ALIGN   F377
B       00CC W        B1      0050 W        B1ADDR 0012 W        B2      0060 W
B2ADDR  0014 W        BFUN    00F4 W*       BFUNH  00F5 M        BFUNL   00F4 M
CA0LP   F296          CA1LP   F2A4          CA2LP  F2B2          CB1LP   F2C0
CB2LP   F2CE          DIRB    00F2 W        DIRBH  00F3 M*       DIRBL   00F2 M*
DIVBY   018E W*       DIVBYH  018F M        DIVBYL 018E M*       DLYLP1  F3A2
DLYLP2  F3B1          ENIR    00D0 M        FILTER F200          FLOOP   F206
INCRM   0200          INIT    F260          INPUT  F309          IRCD    00D4 M
IRPD    00D2 M        K       00CA W        M0     0070 W        M0ADDR  0016 W
M1      0080 W        M1ADDR  0018 W        M2     0090 W        M2ADDR  001A W
MTEMP   000A W        MUAL    205F          MWDONE F310          NCNT    0006W
NOTDN   F30B          NSTG    0004 W        ODONE  F37E          OPOS    F372
OUTPUT  F367          PC      00C6 W        PMORE  F413          PORTB   00E2 W
PORTBH  00E3 M*       PORTBL  00E2 M*       PORTI  0008 M*       PRECOM  F3B8
PRELP   F3BB          PSW     00C0 M*       PTEMP  0008 W        ROMA0   F238
ROMA1   F240          ROMA2   F248          ROMB1  F250          ROMB2   F258
ROMNST  F236          RVAL    E0A1          SIO    00D6 M        SMULT   F20F
SP      00C4 W        SVAL    2100          T1     00A0 W        T1ADDR  001C W
T2      00B0 W        T2ADDR  001E W        T2REG  0186 W        T2TIM   0188 W
TMMD    0190 W        TMMDH   0191 M*       TMMDL  0190 M*       X       00CE W
YCOMP   F325          YLOOP   F328          YMORE  F355          YOFK    0002 W
YOUT    0000 M        ZEROLP  F2D7
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  12
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER
MACRO TABLE

MUTBL
   NO WARNING LINES

   NO ERROR LINES

 1079 ROM BYTES USED

SOURCE CHECKSUM = 4769
OBJECT CHECKSUM = 1378

INPUT   FILE C:FILTER.MAC
LISTING FILE C:FILTER.PRN
OBJECT  FILE C:FILTER.LM
```

A Floating Point Package for the HPC

National Semiconductor
Application Note 486
Ashok Krishnamurthy

INTRODUCTION

This report describes the implementation of a Single Precision Floating Point Arithmetic package for the National Semiconductor HPC microcontroller. The package is based upon the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754–1985). However, the package is not a conforming implementation of the standard. The differences between the HPC implementation and the standard are described later in this report.

The following single precision (SP) operations have been implemented in the package.

(1) FADD. Addition of two SP floating point (FLP) numbers.

(2) FSUB. Subtraction of two SP FLP numbers.

(3) FMULT. Multiplication of two SP FLP numbers.

(4) FDIV. Division of two SP FLP numbers.

(5) ATOF. Convert an ASCII string representing a decimal FLP number to a binary SP FLP number.

(6) FTOA. Convert a binary SP FLP number to a decimal FLP number and output the decimal FLP number as an ASCII string.

The report is organized as follows. The next section discusses the representation of FLP numbers. Then, the differences between the HPC implementation and the IEEE/ANSI standard are described. This is followed by a description of the algorithms used in the computations. Appendix A is a User's Manual for the package, Appendix B describes the test data for the package and Appendix C is a listing of the code.

Note that this report assumes that the reader is familiar with the IEEE/ANSI Binary Floating-Point Standard. Please refer to this document for an explanation of the terms used here.

REPRESENTATION OF FLOATING POINT NUMBERS

The specification of a binary floating point number involves two parts: a mantissa and an exponent. The mantissa is a signed fixed point number and the exponent is a signed integer. The IEEE/ANSI standard specifies that a SP FLP number shall be represented in 32 bits as shown in *Figure 1.*

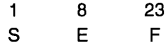


FIGURE 1

The significance of each of these fields is as follows:

1. S—this 1-bit field is the sign of the mantissa. S = 0 means that the number is positive, while S = 1 means that it is negative.
2. E—this is the 8-bit exponent field. The exponent is represented as a biased value with a bias of 127-decimal.
3. F—this is the 23-bit mantissa field. For normalized FLP numbers (see below), a MSB of 1 is assumed and not represented. Thus, for normalized numbers, the value of the mantissa is 1.F. This provides an effective precision of 24 bits for the mantissa.

Normalized FLP number: A binary FLP number is said to be normalized if the value of the MSB of the mantissa is 1. Normalization is important and useful because it provides maximum precision in the representation of the number. If we deal with normalized numbers only (as the HPC implementation does) then since the MSB of the mantissa is always 1, it need not be explicitly represented. This is as specified in the IEEE/ANSI standard.

Given the values of S , E and F, the value of the SP FLP number is obtained as follows.

If 0 < E < 255, then the FLP number is (−1)^S*1.F*2^(E−127).

If E = 0, then the value of the FLP number is 0.

If E = 255, then the FLP number is not a valid number (NAN).

The above format for binary SP FLP numbers provides for the representation of numbers in the range −3.4*10^38 to −1.75*10^−38, 0, and 1.75*10^−38 to 3.4*10^38. The accuracy is between 7 and 8 decimal digits.

DIFFERENCES BETWEEN THE IMPLEMENTATION AND THE IEEE/ANSI STANDARD

The IEEE/ANSI standard specifies a comprehensive list of operations and representations for FLP numbers. Since an implementation that fully conforms to this standard would lead to an excessive amount of overhead, a number of the features in the standard were dropped. This section describes the differences between the implemented package and the standard.

1. Omission of −0. The IEEE/ANSI standard requires that both + and − zero be represented, and arithmetic carried out using both. The implementation does not represent −0. Only +0 is represented and arithmetic is carried out with +0 only.
2. Omission of Infinity Arithmetic. The IEEE/ANSI standard provides for the representation of plus and minus Infinity, and requires that valid arithmetic operations be carried out on Infinity. The HPC implementation does not support this.
3. Omission of Quiet NaN. The IEEE/ANSI standard provides for both quiet and signalling NaNs. The HPC implementation provides for signalling NaNs only. A signalling NaN can be produced as the result of overflow during an arithmetic operation. If the NaN is passed as input to further floating point routines, then these routines will produce another NaN as output. The routines will also set the Invalid Operation flag, and call the user floating point error trap routine at address FPTRAP.
4. Omission of denormalized numbers. Denormalized numbers are FLP numbers with a biased exponent, E of zero and a non zero mantissa F. Such denormalized numbers are useful in providing gradual underflow to zero. Denormalized numbers are not represented or used in the HPC implementation. Instead, if the result of a computation cannot be represented as a normalized number within the allowable exponent range, then an underflow is signaled, the result is set to zero, and the user floating point error trap routine at address FPTRAP is called.
5. Omission of the Inexact Result exception. The IEEE/ANSI standard requires that an Inexact Result exception be signaled when the rounded result of an operation is not exact, or it overflows without an overflow trap. This feature is not provided in the HPC implementation.

6. Biased Rounding to Nearest. The IEEE/ANSI standard requires that rounding to nearest be provided as the default rounding mode. Further, the rounding is required to be unbiased. The HPC implementation provides biased rounding to nearest only. An example will help clarify this.

   Suppose the result of an operation is .b1b2b3XXX and needs to be rounded to 3 binary digits. Then if XXX is 0YY, the round to nearest result is .b1b2b3. If XXX is 1YY, with at least one of the Y's being 1, then the result is .b1b2b3 + 0.001. Finally if XXX is 100, it is a tie situation. In such a case, the IEEE/ANSI standard requires that the rounded result be such that its LSB is 0. The HPC implementation, on the other hand, will round the result in such a case to .b1b2b3 + 0.001.

DESCRIPTION OF ALGORITHMS

1. **General Considerations.** The HPC implementation of the SP floating point package consists of a series of subroutines. The subroutines have been designed to be compatible with the CCHPC C Cross Compiler. They have, however, not been tested with the CCHPC Cross Compiler.

   The Arithmetic subroutines that compute F1 op F2 (where op is +, −, * or /) expect that F1 and F2 are input in the IEEE format. Each of F1 and F2 consists of two 16-bit words organized as follows.

   Fn−HI: S EXP 7 MS bits of F

   Fn−LO: 16 LS bits of F

   In the above, S is the sign of the mantissa, EXP is the biased exponent, and F is the mantissa.

   On input it is assumed that F1−HI is in register K, F1−LO is in the accumulator A, and F2−HI and F2−LO are on the stack just below the return address i.e., F2−HI is at W(SP-4) and F2−LO is at W(SP-6). The result, C, is also returned in IEEE format with C−HI in register K and C−LO in the accumulator A.

   The two Format Conversion routines, ATOF and FTOA expect that on entry, register B contains the address of the start of the ASCII byte string representing the decimal FLP number. ATOF reads the byte string starting from this address. Note that the string must be terminated with a null byte. The binary floating point number is returned in registers K and A. FTOA, on the other hand, writes the decimal FLP string starting from the address in register B on entry. A terminating null byte is also output. Also, FTOA expects that the binary FLP number to be converted is in registers K and A on entry.

   Most of the storage required by the subroutines is obtained from the stack. Two additional words of storage in the base page are also used. The first is W(0), and is referenced in the subroutines as W(TMP1). The second word of storage can be anywhere in the base page and is used to store the sticky flags used to signal floating point exceptions. This is referenced in the subroutines as W(FPERWD). Thus any user program that uses the floating point package needs to have the symbols TMP1 and FPERWD defined appropriately.

2. **Exception Handling.** The following types of exception can occur during the course of a computation.

   (i) Invalid Operand. This exception occurs if one of the input operands is a NaN.

   (ii) Exponent Overflow. This occurs if the result of a computation is such that its exponent has a biased value of 255 or more.

   (iii) Exponent Underflow. This occurs if the result of a computation is such that its exponent is 0 or less.

   (iv) Divide-by-zero. This exception occurs if the FDIV routine is called with F2 being zero.

   The package signals exceptions in two ways. First a word at address FPERWD is maintained that records the history of these exception conditions. Bits 0−3 of this word are used for this purpose.

   Bit 0—Set on Exponent Overflow.

   Bit 1—Set on Exponent Underflow.

   Bit 2—Set on Illegal Operand.

   Bit 3—Set on Divide-by-zero.

   These bits are never cleared by the floating point package, and can be examined by the user software to determine the exception conditions that occurred during the course of a computation. It is the responsibility of the user software to initialize this word before calling any of the floating point routines.

   The second method that the package uses to signal exceptions is to call a user floating point exception handler subroutine whenever an exception occurs. The corresponding exception bit in FPERWD is set before calling the handler. The starting address of the handler should be defined by the symbol FPTRAP.

3. **Unpacked Floating Point Format.** The IEEE/ANSI standard floating point format described earlier is very cumbersome to deal with during computation. This is primarily because of the splitting of the mantissa between the two words. The subroutines in the package unpack the input FLP numbers into an internal representation, do the computations using this representation, and finally pack the result into the IEEE format before return to the calling program. The unpacking is done by the subroutine FUNPAK and the packing by the subroutine FPAK. The unpacked format consists of 3 words and is organized as follows.

Fn-EXP.Fn-SIGN	8 bits biased exponent	sign (extended to 8 bits)
Fn-HI	MS 16 bits of mantissa (implicit 1 is present as MSB)	
Fn-LO	LS 8 bits of mantissa	Eight Zeros

   Since all computations are carried out in this format, note that the result is actually known to 32 bits. This 32-bit mantissa is rounded to 24 bits before being packed to the IEEE format.

4. **Algorithm Description.** All the arithmetic algorithms first check for the easy cases when either F1 or F2 is zero or a NaN. The result in these cases is immediately available. The description of the algorithms below is for those cases when neither F1 nor F2 is zero or a NaN. Also, in order to keep the algorithm description simple, the check for underflow/overflow at the various stages is not shown. The documentation in the program, the descriptions given below, and the theory as described in the references should allow these programs to be easily maintained.

   (i) FADD.

   The processing steps are as follows:

   1. Compare F1-EXP and F2-EXP. Let the difference be D. Shift right the mantissa (Fn-HI.Fn-LO, n = 1 or 2) of the FLP number with the smaller exponent D times. Let the numbers after this step be F1-EXP.F1-SIGN, F1-HI, F1-LO and F2-EXP.F2-SIGN,

F2-HI and F2-LO. This step equalizes the two exponents.

2. Take the XOR of F1-SIGN and F2-SIGN. If this is 0, then go to step 4, else go to step 3.
3. Do a true subtract of F2-LO from F1-LO. (A true subtract is when the SUBC instruction is preceded by a SET C instruction.) Then do a 1's complement subtract of F2-HI from F1-HI. If the last subtract resulted in C = 1, then go to step 3.2, else go to step 3.1.

   3.1. Get here means that F2 is larger than F1, and the computed result is negative. Take the 2's complement of the result to make it positive. Set the sign of the result to be the sign of F2. Go to step 3.3.

   3.2. Get here means F1 is larger than F2, and the result of the mantissa subtract is positive. Set the sign of the result to be the sign of F1. Go to step 3.3.

   3.3. The result after a subtract need not be normalized. Shift left the result mantissa until its MSB is 1. Decrement the exponent of the result by 1 for each such left shift. Go to step 5.
4. Add F2-LO to F1-LO. Next add with any carry from the previous add, F2-HI to F1-HI. If this last add results in C = 1, then go to step 4.1, else go to step 5.

   4.1. Rotate Right with carry C-HI. Next load C-LO in and rotate it right with carry. Increase the exponent of the result, C by 1. Go to step 5.
5. Round the result. Go to step 6.
6. Pack the result and return.

(ii) FSUB.

The processing steps are as follows:

1. Copy F2 to the stack and change its sign. Go to step 2.
2. Call FADD.
3. Remove the copy of -F2 from the stack and return.

(iii) FMULT.

The processing steps are as follows.

1. Add F1-EXP and F2-EXP to get C1-EXP. Subtract from C1-EXP 127-decimal which is the IEEE bias, to get C-EXP. Go to step 2.
2. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 3.
3. Compute F1-HI*F2-HI. Let the upper half of the product be C1-HI and the lower half C1-LO. Go to step 4.
4. Compute F1-HI*F2-LO. Let the upper half of this product be C2-HI. Add C2-HI to C1-LO to give C11-LO. If this last add results in C = 1, then increment C1-HI. Go to step 5.
5. Compute F1-LO*F2-HI. Let the upper half of this product be C3-HI. Add C3-HI to C11-LO to get C12-LO. If this last add results in C = 1, then increment C1-HI. Go to step 6.
6. Mantissa normalization. If the MSB of C1-HI is 1, then increment C-EXP, else shift left C1-HI.C12-LO. Go to step 7.
7. Round C1-HI.C12-LO to get C-HI.C-LO. Go to step 8.
8. Pack C-EXP.C-SIGN, C-HI and C-LO and return as the answer.

(iv) FDIV.

The processing steps are as follows:

1. Compare F1-HI and F2-HI. If F2-HI is greater than F1-HI then go to Step 3, else go to step 2.
2. Shift right F1-HI.F1-LO. Increase F1-EXP by 1.
3. Subtract F2-EXP from F1-EXP. Add to the result 127-decimal to get C1-EXP. Go to step 4.
4. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 5.
5. Compute F1-HI*F2-LO. Let the result be M1-HI.M1-LO. Go to step 6.
6. Divide M1-HI.M1-LO by F2-HI. Let the quotient be M2-HI. Go to step 7.
7. Do a true subtract of M2-HI from F1-LO. Let the result be M3-LO. If C = 1 as a result of this subtract, then go to step 8, else decrement F1-HI and go to step 8.
8. Divide F1-HI.M3-LO by F2-HI. Let the quotient be C1-HI and the remainder R1. Go to step 9.
9. Divide R1 .0000 by F2-HI. Let the quotient be C1-LO. Go to step 10.
10. If the MSB of C1-HI is 1 then go to step 11, else shift left C1-HI.C1-LO, decrease C1-EXP by 1 and go to step 11.
11. Round C1-HI.C1-LO to get C-HI.C-LO. go to step 12.
12. Pack C1-EXP.C-SIGN, C-HI and C-LO and return as the result.

(v) ATOF.

The processing steps in this case are as follows.

1. Set M-SIGN, the mantissa sign to 0.

   Set M10-EXP, the implicit decimal exponent to 0.

   Set HI-INT to 0.

   Set LO-INT to 0.

   Go to step 2.
2. Get a character from the input string. Let the character be C.

   If C is a '+', then go to the start of step 2.

   If C is a '−', then set M-SIGN to FF and go to start of step 2.

   If C is a '.', then go to step 5.

   If C is none of the above, then go to step 3.
3. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 3.1, else go to step 3.2.

   3.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Go to step 4.

   3.2. Increase M10-EXP by 1 and go to step 4.
4. Get a character from the input string. Let the character be C.

   If C is a '.', then go to step 5.

   If C is a 'E', then go to step 7.

   If C is the space character, then go to the start of step 4.

   If C is none of the above, then go to step 3.

5. Get a character from the input string. Let the character be C.

   If C is a 'E', then go to step 7.

   If C is the space character, then go to the start of step 5.

   If C is none of the above, then go to step 6.

6. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 6.1, else go to step 5.

   6.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Decrement M10-EXP by 1. Go to step 5.

7. Set SEXP, the exponent sign to be 0. Go to step 8.

8. Get a character from the input string. Let the character be C.

   If C is a '+', then go to start of step 8.

   If C is a '−', then set SEXP to be FF and go to the start of step 8.

   If C is none of the above, then go to step 9.

9. Set M20-EXP, the explicit decimal exponent to 0. Go to step 10.

10. Subtract 30 from C to get its integer value. Let this be I. Multiply M20-EXP by 10 and add I to the product. Store this sum back in M20-EXP. Go to step 11.

11. Get a character from the input string. Let this be C. If C is the null character, then go to step 12, else go to step 10.

12. Add M10-EXP and M20-EXP (with the proper sign as determined by SEXP) to get the 10's exponent M-EXP. Save in M-EXP the magnitude of the sum and in SEXP the sign of the sum. Go to step 13.

13. Check and see if HI-INT.LO-INT is 0. If it is, then set the resulting floating point number, C, to zero and return. If it is not then go to step 14.

14. Normalize HI-INT.LO-INT by left shifts such that the MSB is 1. Let the number of left shifts needed to do this be L. Set B1-EXP to 32-decimal − L. Go to step 15.

15. If SEXP is 0, then set P-HI.P-LO to the binary representation of 0.625, else set P-HI.P-LO to the binary representation of 0.8. Go to step 16.

16. Multiply HI-INT.LO-INT by P-HI.P-LO M-EXP times. After each multiplication, normalize the partial product if needed by left shifting. Accumulate the number of left shifts needed in B2-EXP. Let the final product be C-HI.C-LO. Go to step 17.

17. Subtract B2-EXP from B1-EXP. Let the result be B-EXP. Go to step 18.

18. If SEXP is 0, then multiply M-EXP by 4, else multiply M-EXP by −3. Let the result be B3-EXP. Go to step 19.

19. Add B-EXP and B3-EXP. Let the result be C1-EXP. Add 126 to C1-EXP to restore the IEEE bias, getting C-EXP. Go to step 20.

20. Round C-HI.C-LO. Go to step 21.

21. Pack C-EXP.M-SIGN, C-HI and C-LO and return.

(vi) FTOA.

The processing steps are as follows.

1. Unpack the input FLP number. Let the unpacked number be represented by C-EXP.C-SIGN, C-HI and C-LO. Go to step 2.

2. Subtract 126-decimal from C-EXP to remove the IEEE bias. Let the result be C1-EXP. Go to step 3.

3. Multiply C1-EXP by the binary representation of log(2). Let the product be U-HI.U-LO. Go to step 4.

4. Subtract 8 from U-HI.U-LO. Let the magitude of the integer part of the result be V and its sign VSIGN. Go to step 5.

5. If VSIGN is 0, then set P-HI.P-LO to the binary representation of 0.8, else set P-HI.P-LO to the binary representation of 0.625. Go to step 6.

6. Multiply C-HI.C-LO by P-HI.P-LO V times. Normalize the partial product after each multiplication, if needed, by left shifting. Accumulate any left shifts needed in B1-EXP. Let the final product be HI-INT.LO-INT. Go to step 7.

7. Subtract B1-EXP from C1-EXP. Let the result be B2-EXP. Go to step 8.

8. If VSIGN is 0, then multiply V by −3, else multiply it by 4. Let the result be B3-EXP. Go to step 9.

9. Add B2-EXP and B3-EXP. Let the result be B4-EXP. Go to step 10.

10. If B4-EXP is more than 32-decimal, then increase V and go to step 6, else go to step 11.

11. If B4-EXP is less than 28-decimal, then decrease V and go to step 6, else go to step 12.

12. Subtract B4-EXP from 32. Let the result be B5-EXP. Go to step 13.

13. Shift HI-INT.LO-INT right B5-EXP number of times. Go to step 14.

14. Add 16-decimal to the address of the start of the decimal string. Output a null byte there. Go to step 15.

15. Divide V by 10-decimal. Let the quotient be Q and the remainder R. Add 30 to R and output it to the decimal string. Next add 30 to Q and output it to the decimal string. Go to step 16.

16. If VSIGN is 0, then output '+' to the output string, else output '−' to the output string. Go to step 17.

17. Output 'E' to the output string. Output '.' to the output string. Go to step 18.

18. Divide C-HI.C-LO by 10-decimal 10 times. Let the remainder in each division be R. Add 30 to each R and output it to the output string. Go to step 19.

19. If C-SIGN is 0, then output the space character to the output string, else output '−' to the output string. Then return to the calling program.

REFERENCES

1. ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, Aug. 12, 1985.
2. J.T. Coonen, "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic," IEEE Computer, Jan. 1980, pp. 68–79.
3. K. Hwang, *Computer Arithmetic,* John-Wiley and Sons, 1979.
4. M. M. Mano, *Computer System Design,* Prentice-Hall, 1980.

APPENDIX A

A USER'S MANUAL FOR THE HPC FLOATING POINT PACKAGE

The Single Precision Floating Point Package for the HPC implements the following functions.

ARITHMETIC FUNCTIONS

1. FADD—Add two floating point numbers.
2. FSUB—Subtract two floating point numbers.
3. FMULT—Multiply two floating point numbers.
4. FDIV—Divide two floating point numbers.

FORMAT CONVERSION FUNCTIONS

5. ATOF—Convert an ASCII string representing a decimal floating point number to a single precision floating point number.
6. FTOA—Convert a single precision floating point number to an ASCII string that represents the decimal floating point value of the number.

The entire package is in the form of a collection of subroutines and is contained in the following files.

1. FERR.MAC
2. FNACHK.MAC
3. FZCHK.MAC
4. FUNPAK.MAC
5. FPAK.MAC
6. FPTRAP.MAC
7. ROUND.MAC
8. BFMUL.MAC
9. ISIOK.MAC
10. MUL10.MAC
11. ATOF.MAC
12. FTOA.MAC
13. FADD.MAC
14. FMULT.MAC
15. FDIV.MAC

The first 7 files are general utility routines that are used by all the Arithmetic and Format Conversion subroutines. The next 3 files, BFMUL.MAC, ISIOK.MAC and MUL10.MAC are used only by the Format Conversion subroutines, ATOF and FTOA. Depending on the functions being used in the user program, only the necessary files need be included.

INTERFACE WITH USER PROGRAMS

1. All the Arithmetic routines expect the input to be in the IEEE Single Precision format. This format requires 2 words for the storage of each floating point number. If the required arithmetic operation is FlopF2, where op is +, −, * or /, then the routines expect that F1 is available in registers K and A on entry, with the high half in K. Also, the two words of F2 are expected to be on the stack. If SP is the stack pointer on entry into one of the Arithmetic function subroutines, then the high word of F2 should be at W(SP-4) and the low word at W(SP-6). The result of the Arithmetic operation is returned in IEEE format in registers K and A, with the high word in K.
2. The Format Conversion subroutine ATOF expects that on entry, B contains the address of the ASCII string representing the decimal floating point number. This string must be of the form

   Siiiii.ffffffEsNND

   where

   S is an optional sign for the mantissa. Thus S can be '+', '−' or not present at all.

   iiiii is the optional integer part of the mantissa. If it is present, it can be of any length, must contain only the characters '0' through '9' and must not contain any embedded blanks.

   . is the optional decimal point. It need not be present if the number has no fractional part.

   ffffff is the optional fractional part of the mantissa. ffffff, if it is present must consist of a sequence of digits '0' through '9'. It can be of any length. Note that either iiiii, the integer part or .ffffff the fractional part must be present.

   E is the required exponent start symbol.

   s is the optional sign of the exponent. If it is present, it must be '+' or '−'.

   NN is the exponent and consists of at most two decimal digits. It is required to be present.

   D is the null byte <00> and must be present to terminate the string.

   The floating point number represented by the above string is returned by ATOF in IEEE format in registers K and A.
3. The format conversion routine FTOA expects the floating point number input to be in registers K and A in the IEEE format. Register B is expected to contain the starting address of a 17 byte portion of memory where the output string will be stored.
4. Three global symbols need to be defined in the user program before assembling the user program and any included floating point package files. These symbols are:
   (i) TMP1 which must be set to 0. The package uses W(TMP1) for temporary storage.
   (ii) FPERWD which must be set to an address in the base page. The package signals floating point exceptions using W(FPERWD). This is described below.
   (iii) FPTRAP which must be set to the address of the start of a user floating point exception handler. Again this is described below.

FLOATING POINT EXCEPTS

The package maintains a history of floating point exceptions in the 4 least significant bits of the word W(FPERWD). The value of the symbol FPERWD should be defined by the user program, and should be an address in the base page. This word should also be cleared by the user program before calling any floating point routine. The word is never cleared by the floating point package, and the user program can examine this word to determine the type of exceptions that may have occurred during the course of a computation.

The following 4 types of error can occur in the course of a floating point computation.

1. Invalid Operand. This happens if one of the input numbers for an Arithmetic routine or the input for FTOA is not a valid floating point number. An invalid floating point number (or NaN) can be created either by an overflow in a previous computation step, or if the ASCII decimal floating point number input to ATOF is too large to be represented in the IEEE format. The result, if one of the inputs is a NaN is always set to a NaN.
2. Overflow. This happens if the result of a computation is too large to be represented within the exponent range available. Overflow can occur in any of the arithmetic routines or ATOF. On overflow, the result is set to a representation called NaN. An NaN is considered an illegal operand in all successive steps.
3. Underflow. This occurs if the result of a computation is too small to be represented with the precision and exponent range available. On underflow, the result is set to zero.
4. Divide-by-zero. This error occurs if F2 is zero when computing F1/F2. The result is set to an NaN.

Each of the above errors results in a bit being set in W(FPERWD). This is done as follows:

Bit 0—Set on Overflow.

Bit 1—Set on Underflow.

Bit 2—Set on Illegal Operand.

Bit 3—Set on Divide-by-zero.

One further action is taken when a floating point exception occurs. After the result has been set to the appropriate value, and the corresponding bit in W(FPERWD) set, the package does a subroutine call to address FPTRAP. The user can provide any exception handler at this address. The file FPTRAP.MAC contains the simplest possible user exception handler. It does nothing, but merely returns back to the calling program.

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP

   1                                        .TITLE FLP
   2                      LISTER:
   3     0071                               .LIST 071
   4     F000                               . = 0F000
   5     0002                               FPERWD = W(2)
   6     0000                               TMP1 = W(0)

NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     2
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
THE FLP ROUTINES

   7                                        .FORM 'THE FLP ROUTINES'
```

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
FERR.MAC

                                        .FORM 'FERR.MAC'
                                        .INCLD FERR.MAC
                        ; EXCEPTION HANDLING.
                        ; DIVIDE BY ZERO.
                        DIVBYO:
F000 820802FA                   OR FPERWD, 08    ; SET THE DIVIDE BY 0 BIT.
F004 00                         CLR A
F005 B17F80                     LD K, 07F80
F008 3093                       JSR FPTRAP
F00R 3FCC                       POP B
F00C 3FCE                       POP X
F00E 3C                         RET
                        ; ILLEGAL OPERAND - ONE OF F1 OR F2 IS A NAN.
                        FNAN:
F00F 820402FA                   OR FPERWD, 04    ; SET THE ILLEGAL OPERAND BIT.
F013 00                         CLR A
F014 B17F80                     LD K, 07F80      ; RETURN NAN IN K AND A.
F017 3084                       JSR FPTRAP       ; GO TO USER TRAP ROUTINE.
F019 3FCC                       POP B
F01B 3FCE                       POP X
F01D 3C                         RET
                        ; EXPONENT UNDERFLOW.
                        UNDFL:
F01E 820202FA                   OR FPERWD, 02    ; SET THE EXPONENT UNDERFLOW BIT.
F022 00                         CLR A
F023 ACC8CA                     LD K, A
F026 3075                       JSR FPTRAP
F028 3FC4                       POP SP
F02A 3FCC                       POP B
F02C 3FCE                       POP X
F02E 3C                         RET
                        ; EXPONENT OVERFLOW.
                        OVRFL:
F02F 820102FA                   OR FPERWD, 01    ; SET THE EXPONENT OVERFLOW BIT.
F033 00                         CLR A
F034 B17F80                     LD K, 07F80
F037 3064                       JSR FPTRAP
F039 3FC4                       POP SP
F03B 3FCC                       POP B
F03D 3FCE                       POP X
F03F 3C                         RET
                        ;
                                .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
FNACHK.MAC

  10                                    .FORM 'FNACHK.MAC'
  11                                    .INCLD FNACHK.MAC
   1                                    .TITLE FNACHK
   2                                    .LOCAL
   3                        ;
   4                        ; SUBROUTINE TO CHECK IF A SP FLOATING POINT NUMBER STORED IN THE
   5                        ; IEEE FLOATING POINT FORMAT IN REGS. K AND A IS NAN.
   6                        ;
   7                        ; RETURNS 0 IN C IF THE NUMBER IS NOT A NAN.
   8                        ; RETURNS 1 IN C IF THE NUMBER IS A NAN.
   9                        ;
  10                        ; PRESERVES REGS. K, A, X AND B. DESTROYS C.
  11                        ;
  12                        FNACHK:
  13 F040 AECA                      X A, K
  14 F042 E7                        SHL A
  15 F043 BDFEFF                    IFGT A,0FEFF
  16 F046 45                        JP $ISNAN
  17 F047 D7                        RRC A
  18 F048 03                        RESET C
  19 F049 AECA                      X A, K
  20 F04B 3C                        RET
  21                        $ISNAN:
  22 F04C D7                        RRC A
  23 F04D 02                        SET C
  24 F04E AECA                      X A, K
  25 F050 3C                        RET
  26                        ;
  27                                .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FNACHK
FZCHK.MAC

 12                                    .FORM 'FZCHK.MAC'
 13                                    .INCLD FZCHK.MAC
  1                                    .TITLE FZCHK
  2                                    .LOCAL
  3                    ;
  4                    ; SUBROUTINE THAT CHECKS IF A SP FLOATING POINT NUMBER STORED
  5                    ; IN THE IEEE FORMAT IN REGS K AND A IS ZERO.
  6                    ;
  7                    ; RETURNS 0 IN C IF THE NUMBER IS NOT ZERO.
  8                    ; RETURNS 1 IN C IF THE NUMBER IS ZERO.
  9                    ; SAVES REGS. K, A, X, AND B BUT DESTROYS C.
 10                    ;
 11                    FZCHK:
 12 F051 AECA                 X A, K
 13 F053 E7                   SHL A
 14 F054 9DFF                 IFGT A,0FF
 15 F056 45                   JP $ANOTO
 16 F057 D7                   RRC A
 17 F058 02                   SET C
 18 F059 AECA                 X A, K
 19 F05B 3C                   RET
 20                    $ANOTO:
 21 F05C D7                   RRC A
 22 F05D 03                   RESET C
 23 F05E AECA                 X A, K
 24 F060 3C                   RET
 25                    ;
 26                                    .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FZCHK
FUNPAK.MAC

  14                                    .FORM 'FUNPAK.MAC'
  15                                    .INCLD FUNPAK.MAC
   1                                    .TITLE FUNPAK
   2                                    .LOCAL
   3                        ;
   4                        ; SUBROUTINE TO UNPACK A SP FLOATING POINT NUMBER STORED IN THE
   5                        ; IEEE FORMAT IN REGS. K AND A. THE UNPACKED FORMAT OCCUPIES 3
   6                        ; WORDS AND IS ORGANIZED AS FOLLOWS:
   7                        ;                          |                |
   8                        ; increasing addrs |       |                |          <- X on exit
   9                        ;                          |EEEEEEEESSSSSSSS| FEXP-FSIGN
  10                        ;                          |MMMMMMMMMMMMMMMM| FHI
  11                        ;                          |MMMMMMMM00000000| FLO      <- X on entry
  12                        ;                          |                |
  13                        ;
  14                        ; EEEEEEEE - 8 BIT EXPONENT IN EXCESS-127 FORMAT
  15                        ; SSSSSSSS - SIGN BIT < 00 -> +, FF -> ->
  16                        ; M ... M - 24 BITS OF MANTISSA. NOTE THAT IMPLIED 1 IS PRESENT HERE.
  17                        ;
  18                        ; ON ENTRY TO THE SUBROUTINE X SHOULD POINT TO FLO. ON EXIT, X POINTS
  19                        ; TO THE WORD AFTER FSIGN.
  20                        ; REGS. K, A AND B ARE DESTROYED BY THIS SUBROUTINE.
  21                        ;
  22                        FUNPAK:
  23 F061 ABCC                      ST A,B          ; SAVE A IN B.
  24 F063 00                        CLR A
  25 F064 D1                        X A, M(X+)      ; ZERO LOW BYTE OF FLO.
  26 F065 88CC                      LD A, L(B)
  27 F067 D1                        X A, M(X+)      ; MOVE LOW BYTE OF F-RO INTO HIGH BYTE OF FLO.
  28 F068 88CD                      LD A, H(B)
  29 F06A D1                        X A, M(X+)      ; MOVE MID BYTE OF MANT INTO LOW BYTE OF FHI.
  30 F06B A8CA                      LD A, K
  31 F06D 96C80F                    SET A.7         ; SET IMPLIED 1 IN MANTISSA
  32 F070 D1                        X A, M(X+)      ; MOVE HIGH BYTE OF MANT INTO HIGH BYTE OF FHI.
  33 F071 A8CA                      LD A, K
  34 F073 E7                        SHL A           ; SIGN BIT TO CARRY.
  35 F074 B9FF00                    AND A, 0FF00    ; ZERO SIGN.
  36 F077 07                        IF C
  37 F078 9AFF                      OR A, 0FF       ; PUT SIGN BACK IF -.
  38 F07A F1                        X A, W(X+)      ; SAVE FEXP-FSIGN.
  39 F07B 3C                        RET
  40                        ;
  41                                .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FUNPAK
FPAK.MAC

  16                                    .FORM 'FPAK.MAC'
  17                                    .INCLD FPAK.MAC
   1                                    .TITLE FPAK
   2                                    .LOCAL
   3                    ;
   4                    ; SUBROUTINE TO PACK A SP FLOATING POINT NUMBER STORED IN THE
   5                    ; 3 WORD FEXP-FSIGN/FHI/FLO FORMAT INTO THE IEEE FORMAT IN REGS.
   6                    ; K AND A.
   7                    ;
   8                    ; ON ENTRY TO THE SUBROUTINE, X POINTS TO FLO. ON EXIT, X POINTS
   9                    ; TO THE WORD AFTER FSIGN.
  10                    ;
  11                    ; REGS. K, A AND B ARE DESTROYED.
  12                    ;
  13                    FPAK:
  14 F07C D1                    X A, M(X+)      ; GET RID OF ZERO LOW BYTE OF FLO.
  15 F07D D1                    X A, M(X+)      ; GET HIGH BYTE OF FLO.
  16 F07E ABCA                  ST A, K         ; STORE IT IN K.
  17 F080 D1                    X A, M(X+)      ; GET LOW BYTE OF FHI.
  18 F081 3B                    SWAP A
  19 F082 3B                    SWAP A
  20 F083 B9FF00                AND A, 0FF00    ; SHIFT LEFT 8 TIMES.
  21 F086 A0C8CAFA              OR K, A         ; LOW WORD OF RESULT IS NOW IN K.
  22                    ;
  23 F08A D1                    X A, M(X+)      ; GET HIGH BYTE OF FHI.
  24 F08B 96C81F                RESET A.7       ; ZERO IMPLIED MSB 1 IN MANT.
  25 F08E ABCC                  ST A,B          ; SAVE IN REG. B.
  26 F090 D4                    LD A, M(X)      ; GET SIGN BYTE FROM ASIGN.
  27 F091 C7                    SHR A           ; MOVE 1 SIGN BIT INTO CARRY.
  28 F092 F0                    LD A, W(X+)     ; GET FEXP-FSIGN.
  29 F093 B9FF00                AND A, 0FF00    ; ZERO SIGN.
  30 F096 D7                    RRC A           ; MOVE RIGHT 1 BIT. SIGN BIT FROM C
  31                                            ; ENTERS INTO THE MSB.
  32 F097 96CCFA                OR A, B         ; GET MANT BITS IN FROM B.
  33 F09A AECA                  X A, K          ; SWAP A AND K
  34 F09C 3C                    RET
  35                    ;
  36                                    .END


NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     8
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPAK
FPTRAP.MAC

  18                                    .FORM 'FPTRAP.MAC'
  19                                    .INCLD FPTRAP.MAC
   1                                    .TITLE FPTRAP
   2                    ; USER SUPPLIED FP TRAP ROUTINE.
   3                    FPTRAP:
   4 F09D 3C                    RET
   5                    ;
   6                                    .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     9
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPTRAP
ROUND.MAC

 20                                  .FORM 'ROUND.MAC'
 21                                  .INCLD ROUND.MAC
  1                                  .TITLE SROUND
  2                                  .LOCAL
  3                        ;
  4                        ; THIS SUBROUTINE IS USED TO ROUND THE 32 BIT MANTISSA OBTAINED
  5                        ; IN THE FLOATING POINT CALCULATIONS TO 24 BITS.
  6                        ;
  7                        ; THE UNPACKED FLOATING POINT NUMBER SHOULD BE STORED IN
  8                        ; CONSECUTIVE WORDS OF MEMORY. ON ENTRY, X SHOULD CONTAIN
  9                        ; THE ADDRESS OF C-HI. C-EXP.C-SIGN IS AT W(X+2) AND
 10                        ; C-LO IS AT W(X-2).
 11                        ;
 12                        ; ON EXIT X HAS THE ADDRESS OF C-EXP.C-SIGN.
 13                        SROUND:
 14 F09E F2                       LD A, W(X-)      ; REMEMBER X POINTS TO C-HI.
 15 F09F F4                       LDA, W(X)        ; LOAD C-LO.
 16 F0A0 96C817                   IF A.7           ; IF BIT 25 OF MANTISSA IS 1,
 17 F0A3 43                       JP $RNDUP        ; THEN NEED TO INCREASE MANTISSA.
 18 F0A4 F0                       LD A, W(X+)
 19 F0A5 F0                       LD A, W(X+)      ; X NOW POINTS TO C-EXP.C-SIGN.
 20 F0A6 5F                       JP $EXIT         ; DONE, SO GET OUT.
 21                        ; INCREASE MANTISSA.
 22                        $RNDUP:
 23 F0A7 B80100                   ADD A, 0100
 24 F0AA F1                       X A, W(X+)       ; INCREASE LOW BYTE BY 1.
 25 F0AB 07                       IF C             ; IF THERE IS A CARRY,
 26 F0AC 42                       JP $HIUP         ; THEN NEED TO INCREASE C-HI.
 27 F0AD F0                       LD A, W(X+)      ; X NOW POINTS TO C-EXP.C-SIGN.
 28 F0AE 57                       JP $EXIT         ; DONE, SO GET OUT.
 29                        ; MANTISSA INCREASE PROPAGATING TO HIGH WORD.
 30                        $HIUP:
 31 F0AF F4                       LD A, W(X)
 32 F0B0 B8                       .BYTE 0B8,00,01 ; DO ADD A, 01 BUT WITH WORD CARRY!!
    F0B1 00
    F0B2 01
 33 F0B3 07                       IF C             ; IF THERE IS A CARRY,
 34 F0B4 42                       JP $EXIN2        ; THEN NEED TO INCREASE EXPONENT.
 35 F0B5 F1                       X A,W(X+)
 36 F0B6 4F                       JP $EXIT         ; GET OUT.
 37                        ; ROUND UP LEADS TO EXPONENT INCREASE.
 38                        $EXIN2:
 39 F0B7 D7                       RRC A            ; CARRY->MSB, LSB->CARRY.
 40 F0B8 F3                       X A,W(X-)
 41 F0B9 F4                       LD A,W(X)        ; LOW WORD IS NOW IN A.
 42 F0BA D7                       RRC A
 43 F0BB F1                       X A, W(X+)
 44 F0BC F0                       LD A, W(X+)      ; X NOW POINTS TO C-EXP.CSIGN.
 45 F0BD F4                       LD A, W(X)
 46 F0BE B80100                   ADD A, 0100
 47 F0C1 07                       IF C
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    10
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
ROUND.MAC

  48 F0C2 BAFF00                OR A, 0FF00     ; MAKE IT A NAN.
  49 F0C5 F6                    ST A, W(X)
  50                    ;
  51                    $EXIT:
  52 F0C6 3C                    RET
  53                    ;
  54                              .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    11
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
BFMUL.MAC

 22                                    .FORM 'BFMUL.MAC'
 23                                    .INCLD BFMUL.MAC
  1                                    .TITLE BFMUL
  2                    ;
  3                    ; THIS SUBROUTINE IS USED TO MULTIPLY TWO 32 BIT FIXED POINT FRACTIONS.
  4                    ; THE ASSUMED BINARY POINT IS TO THE IMMEDIATE LEFT OF THE MSB.
  5                    ;
  6                    ; THE FIRST FRACTION IS STORED IN REGS K AND A, WITH THE MORE
  7                    ; SIGNIFICANT WORD BEING IN K.
  8                    ;
  9                    ; THE SECOND FRACTION IS STORED ON THE STACK. THE MORE SIGNIFICANT
 10                    ; WORD IS AT W(SP-4) AND THE LOWER SIGNIFICANT WORD
 11                    ; IS IN THE WORD BELOW IT.
 12                    ;
 13                    ; THE 32 BIT PRODUCT IS LEFT IN REGS. K AND A, WITH THE MORE
 14                    ; SIGNIFICANT WORD BEING IN K.
 15                    ;
 16                    ; IMPORTANT NOTE : THE FRACTIONS ARE ASSUMED TO BE UNSIGNED.
 17                    ;
 18                    ; REGS. B AND X ARE UNCHANGED.
 19                    ;
 20                    BFMUL:
 21 F0C7 AFCE                 PUSH X           ; SAVE X.
 22 F0C9 AFC8                 PUSH A           ; SAVE F1-LO
 23 F0CB AFCA                 PUSH K           ; SAVE F1-HI.
 24 F0CD ABCA                 LD A, K          ; MOVE F1-HI TO A.
 25 F0CF A6FFF6C4FE           MULT A, W(SP-0A) ; MULTIPLY F1-HI BY F2-HI.
 26 F0D4 3FCA                 POP K            ; GET FI-HI.
 27 F0D6 AFCE                 PUSH X           ; SAVE PR-HI.
 28 F0D8 AFC8                 PUSH A           ; SAVE PR-LO.
 29 F0DA A8CA                 LD A, K          ; MOVE F1-HI TO A.
 30 F0DC A6FFF2C4FE           MULT A, W(SP-0E) ; MULTIPLY F1-HI BY F2-LO.
 31 F0E1 3FC8                 POP A            ; GET PR-LO SAVED. NOTE THAT THE
 32                                            ; LO WORD OF THIS PRODUCT IS DISCARDED.
 33 F0E3 3FCA                 POP K            ; GET PR-HI SAVED.
 34 F0E5 96CEF8               ADD A, X         ; ADD TO PR-LO THE HI WORD OF THIS PRODUCT.
 35 F0E8 07                   IF C             ; ON CARRY,
 36 F0E9 A9CA                 INC K            ; PROPAGATE THRU TO PR-HI.
 37 F0EB 3FCE                 POP X            ; GET F1-LO.
 38 F0ED AFCA                 PUSH K           ; SAVE PR-HI.
 39 F0EF AFC8                 PUSH A           ; SAVE PR-LO.
 40 F0F1 A8CE                 LD A, X          ; MOVE F1-LO TO A.
 41 F0F3 A6FFF6C4FE           MULT A, W(SP-0A) ; MULTIPLY BY F2-HI.
 42 F0F8 3FC8                 POP A            ; GET PR-LO SAVED.
 43 F0FA 3FCA                 POP K            ; GET PR-HI SAVED.
 44 F0FC 96CEF8               ADD A, X         ; ADD TO PR-LO THE HI-WORD OF THIS PRODUCT.
 45 F0FF 07                   IF C
 46 F100 A9CA                 INC K            ; PROPAGATE ANY CARRY TO PR-HI.
 47 F102 3FCE                 POP X            ; RESTORE X.
 48 F104 3C                   RET
 49                    ;
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    12
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
BFMUL
BFMUL.MAC

  50                                    .END


NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    13
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
BFMUL
ISIOK.MAC

  24                                    .FORM 'ISIOK.MAC'
  25                                    .INCLD ISIOK.MAC
   1                                    .TITLE ISIOK
   2                                    .LOCAL
   3                    ;
   4                    ; THIS SUBROUTINE IS USED TO DETERMINE IF ANOTHER DECIMAL DIGIT CAN
   5                    ; BE ACCUMULATED IN THE 32 BIT INTEGER STORED IN REGS. K AND A.
   6                    ; THE MORE SIGNIFICANT WORD IS IN K.
   7                    ; SETS THE CARRY TO 1 IF IT CAN BE ACCUMULATED; RESETS THE CARRY
   8                    ; OTHERWISE. PRESERVES ALL REGS.
   9                    ;
  10                    ISIOK:
  11 F105 02                    SET C
  12 F106 861999CAFC            IFEQ K, 01999
  13 F10B 47                    JP $CHKOT
  14 F10C 861999CAFD            IFGT K, 01999
  15 F111 03                    RESET C
  16 F112 3C                    RET
  17 F113 BD9998        $CHKOT: IFGT A, 09998
  18 F116 03                    RESET C
  19 F117 3C                    RET
  20                    ;
  21                                    .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    14
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ISIOK
MUL10.MAC

                                    .FORM 'MUL10.MAC'
                                    .INCLD MUL10.MAC
                                    .TITLE MUL10
                                    .LOCAL
                    ;
                    ; THIS SUBROUTINE MULTIPLIES THE 32 BIT INTEGER STORED IN REGS K AND A
                    ; BY 10-DECIMAL AND ADDS TO IT THE INTEGER STORED IN X.
                    ; THE RESULT IS RETURNED IN K AND A.
                    ; REGS. B AND X ARE NOT CHANGED.
                    ;
                    MUL10:
F118 AFCE                   PUSH X          ; SAVE INTEGER.
F11A AFC8                   PUSH A          ; SAVE LONG INT-LO.
F11C A8CA                   LD A, K
F11E 9E0A                   MULT A, 0A      ; MULT LONG INT-HI BY 10.
F120 AFC8                   PUSH A          ; SAVE LOW WORD OF PRODUCT.
F122 A6FFFCC4A8             LD A, W(SP-4)   ; GET LONG INT-LO.
F127 9E0A                   MULT A, 0A
F129 3FCA                   POP K           ; GET LO WORD OF LAST PRODUCT.
F12B A0CECAF8               ADD K, X        ; ADD TO IT HI WORD OF THIS PRODUCT.
F12F 3FCE                   POP X           ; GET RID OF GARBAGE.
F131 3FCE                   POP X           ; GET INTEGER TO BE ADDED.
F133 96CEF8                 ADD A, X
F136 07                     IF C
F137 A9CA                   INC K
F139 3C                     RET
                    ;
                                .END
                    ;
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    15
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
MUL10
ATOF.MAC

 29                                  .FORM 'ATOF.MAC'
 30                                  .INCLD ATOF.MAC
  1                                  .TITLE ATOF
  2                                  .LOCAL
  3                          ;
  4                          ; THIS SUBROUTINE CONVERTS A DECIMAL FLOATING POINT STRING TO
  5                          ; AN IEEE FORMAT SINGLE PRECISION FLOATING POINT NUMBER. THE
  6                          ; INPUT DECIMAL STRING IS ASSUMED TO BE OF THE FORM
  7                          ;                SMMMMMMM.FFFFFEDNN
  8                          ; WHERE S IS THE SIGN OF THE DECIMAL MANTISSA,
  9                          ;       M...M IS THE INTEGER PART OF THE MANTISSA,
 10                          ;       F...F IS THE FRACTIONAL PART OF THE MANTISSA,
 11                          ;       D IS THE SIGN OF THE DECIMAL EXPONENT,
 12                          ; AND   NNN IS THE DECIMAL EXPONENT.
 13                          ;
 14                          ;       ON ENTRY, B SHOULD POINT TO THE ADDRESS OF THE ASCII
 15                          ; STRING HOLDING THE DECIMAL FLOATING POINT NUMBER. THIS STRING
 16                          ; MUST BE TERMINATED BY A NULL BYTE.
 17                          ;
 18                          ;       THE BINARY FLOATING POINT NUMBER IS RETURNED IN
 19                          ; REGS. K AND A.
 20                          ;
 21                          ; REGS. B AND X ARE LEFT UNCHANGED.
 22                          ;
 23                          ;
 24                          ;
 25                          ;
 26                          ;
 27                          ATOF:
 28 F13A AFCE                        PUSH X
 29 F13C AFCC                        PUSH B
 30 F13E 00                          CLR A           ; ZERO A.
 31 F13F AFC8                        PUSH A          ; STORAGE FOR MANTISSA SIGN.
 32 F141 AFC8                        PUSH A          ; STORAGE FOR IMPLICIT 10'S EXPONENT.
 33 F143 AFC8                        PUSH A          ; STORAGE FOR HI-INT.
 34 F145 AFC8                        PUSH A          ; STORAGE FOR LO-INT.
 35                          ;
 36                          ; DECIMAL STRING MUST START WITH A '+', '-', '.' OR A DIGIT.
 37                          ; RESULTS ARE UNPREDICTABLE IF IT DOES NOT.
 38                          ; THE '+' MEANS THAT THE MANTISSA IS POSITIVE. IT CAN BE OMITTED.
 39                          ; THE '-' MEANS THAT THE MANTISSA IS NEGATIVE.
 40                          ; THE '.' MEANS THAT THE MANTISSA HAS NO INTEGER PART.
 41                          ;
 42                          $LOOP1:
 43 F147 C0                          LDS A, M(B+)
 44 F148 40                          NOP             ; GET THE CHARACTER.
 45 F149 9C2B                        IFEQ A, '+'     ; IF IT IS A '+',
 46 F14B 64                          JP $LOOP1       ; DO NOTHING, BUT GET 1 MORE.
 47 F14C 9C2D                        IFEQ A, '-'     ; IF IT IS A '-',
 48 F14E 45                          JP $MSIGN       ; GO AND CHANGE THE MANTISSA SIGN.
 49 F14F 9C2E                        IFEQ A, '.'     ; IF IT IS A '.',
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    16
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

  50 F151 9438                    JMP $FRCOL      ; GO AND COLLECT THE FRACTION PART.
  51                                              ; GET HERE MEANS IT IS A DIGIT.
  52 F153 48                      JP $INCOL       ; SO GO AND COLLECT THE INTEGER PART.
  53                            $MSIGN:
  54 F154 90FF                    LD A, OFF
  55 F156 A6FFF8C4AB              ST A, W(SP-08)  ; CHANGE MANTISSA SIGN TO NEG.
  56 F15B 74                      JP $LOOP1       ; GO BACK FOR SOME MORE.
  57                            ;
  58                            $INCOL:
  59                            ; GET HERE MEANS COLLECTING INTEGER PART OF MANTISSA.
  60                            ;
  61 F15C 02                      SET C
  62 F15D 8230C8EB                SUBC A, '0'     ; CONVERT DIGIT FROM ASCII TO INTEGER.
  63 F161 ACC8CE                  LD X, A         ; MOVE INTEGER TO X.
  64 F164 3FCA                    POP K           ; GET HI-INT COLLECTED SO FAR.
  65 F166 3FC8                    POP A           ; GET LO-INT COLLECTED SO FAR.
  66 F168 3463                    JSR ISIOK       ; CHECK IF THE DIGIT CAN BE ACCUMULATED.
  67 F16A 07                      IF C            ; LOOK AT C.
  68 F16B 4B                      JP $ACCM        ; YES, IT CAN BE SO GO DO IT.
  69                                              ; GET HERE MEANS CAN ACCUMULATE ANY MORE.
  70                                              ; SO INCREASE THE IMPLICIT 10'S EXPONENT.
  71 F16C 3FCE                    POP X           ; GET IMPLICIT 10'S EXPONENT COLLECTED
  72 F16E A9CE                    INC X           ; SO FAR AND INCREMENT IT.
  73 F170 AFCE                    PUSH X          ; SAVE IT BACK.
  74 F172 AFC8                    PUSH A          ; SAVE LO-INT.
  75 F174 AFCA                    PUSH K          ; SAVE HI-INT.
  76 F176 46                      JP $ISNXT
  77                            ;
  78                            $ACCM:
  79                            ; GET HERE MEANS THE PRESENT DIGIT CAN BE ACCUMULATED.
  80 F177 345F                    JSR MUL10       ; MULTIPLY BY 10 AND ADD DIGIT.
  81 F179 AFC8                    PUSH A          ; SAVE LO-INT.
  82 F17B AFCA                    PUSH K          ; SAVE HI-INT.
  83                            $ISNXT:
  84                            ; PROCESS THE NEXT CHARACTER.
  85 F17D C0                      LDS A, M(B+)
  86 F17E 40                      NOP
  87 F17F 9C2E                    IFEQ A, '.'     ; IF IT IS A '.'
  88 F181 49                      JP $FRCOL       ; GO COLLECT FRACTION PART.
  89 F182 9C45                    IFEQ A, 'E'     ; IF IT IS 'E',
  90 F184 9434                    JMP $EXCOL      ; GO COLLECT EXPONENT PART.
  91 F186 9C20                    IFEQ A, ' '     ; IF IT IS A SPACE,
  92 F188 6B                      JP $ISNXT       ; GO GET SOME MORE.
  93                                              ; GET HERE MEANS IT IS A DIGIT.
  94 F189 952D                    JMP $INCOL
  95                            ;
  96                            $FRCOL:
  97                            ; GET HERE MEANS COLLECT THE FRACTIONAL PART OF THE MANTISSA.
  98                            ;
  99 F18B C0                      LDS A, M(B+)
 100 F18C 40                      NOP
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    17
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

101 F18D 9C45                   IFEQ A, 'E'     ; IF IT IS A 'E',
102 F18F 9429                   JMP $EXCOL      ; GO COLLECT EXPONENT.
103 F191 9C20                   IFEQ A, ' '     ; IF IT IS SPACE,
104 F193 68                     JP $FRCOL       ; GO GET SOME MORE.
105                                             ; GET HERE MEANS IT IS A DIGIT.
106 F194 D2                     SET C
107 F195 8230C8EB               SUBC A, '0'     ; GET INTEGER FROM DIGIT.
108 F199 ACC8CE                 LD X, A         ; SAVE IT IN A.
109 F19C 3FCA                   POP K           ; GET HI-INT.
110 F19E EFC8                   POP A           ; GET LO-INT.
111 F1A0 349B                   JSR ISIOK       ; CHECK IF IT CAN BE ACCUMULATED.
112 F1A2 07                     IF C
113 F1A3 45                     JP $ACCF        ; YES, SO GO DO IT.
114                                             ; GET HERE MEANS CAN'T COLLECT MORE DIGITS.
115 F1A4 AFC8                   PUSH A
116 F1A6 AFCA                   PUSH K
117 F1A8 7D                     JP $FRCOL       ; SO JUST IGNORE IT.
118                     ;
119                     $ACCF:
120                     ; ACCUMULATE THE FRACTIONAL DIGIT.
121 F1A9 3491                   JSR MUL10       ; MULTIPLY BY 10 AND ADD DIGIT.
122 F1AB 3FCE                   POP X           ; GET IMPLICIT 10'S EXPONENT COLLECTED SO FAR,
123 F1AD 86FFFFCEF8             ADD X, OFFFF    ; AND DECREMENT IT BY 1.
124 F1B2 AFCE                   PUSH X          ; SAVE IT BACK.
125 F1B4 AFC8                   PUSH A          ; SAVE LO-INT.
126 F1B6 AFCA                   PUSH K          ; SAVE HI-INT.
127 F1B8 952D                   JMP $FRCOL      ; GO GET SOME MORE.
128                     ;
128                     $EXCOL:
130                     ; GET HERE MEANS THE EXPLICIT 10'S EXPONENT NEEDS TO BE
131                     ; COLLECTED FROM THE STRING.
132 F1BA 03                     RESET C         ; MAKE EXPONENT SIGN POST.
133                     $EXCHR:
134 F1BB C0                     LDS A, M(B+)
135 F1BC 40                     MOP
136 F1BD 9C2B                   IFEQ A, '+'     ; IF IT IS A '+',
137 F1BF 64                     JP $EXCHR       ; GET SOME MORE.
138 F1C0 9C2D                   IFEQ A, '-',    ; IF IT IS A '-',
139 F1C2 44                     JP $ESIGN       ; GO FIX EXPONENT SIGN.
140 F1C3 9C20                   IFEQ A, ' '     ; IF IT IS SPACE,
141 F1C5 6A                     JP $EXCHR       ; GO GET SOME MORE.
142                                             ; GET HERE MEANS IT IS A DIGIT.
143 F1C6 42                     JP $EXACC       ; SO GO COLLECT THE EXPONENT.
144                     $ESIGN:
145 F1C7 02                     SET C
146 F1C8 6D                     JP $EXCHR
147                     ;
148                     $EXACC:
149                     ; ACCUMULATE THE EXPLICIT 10'S EXPONENT.
150 F1C9 9100                   LD K, 0
151 F1CB 07                     IF C
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    18
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

152 F1CC 91FF                   LD K, OFF       ; GET SIGN BITS SET.
153 F1CE AFCA                   PUSH K          ; SAVE EXPLICIT EXPONENTS SIGN.
154 F1D0 9300                   LD X, 0
155 F1D2 AFCE                   PUSH X          ; ZERO EXPLICIT EXPONENT COLLECTED SO FAR.
156                     $EXCLP:
157 F1D4 02                     SET C
158 F1D5 8230C8EB               SUBC A, '0'     ; GET INTEGER FROM ASCII DIGIT.
159 F1D9 ACC8CE                 LD X, A
160 F1DC 3FC8                   POP A           ; GET EXPLICIT EXPONENT COLLECTED SO FAR.
161 F1DE A0C8CEF8               ADD X, A        ; X CONTAINS DIGIT + EXP.
162 F1E2 A0C8CEF8               ADD X, A        ; X CONTAINS DIGIT + 2*EXP.
163 F1E6 E7                     SHL A
164 F1E7 E7                     SHL A
165 F1E8 E7                     SHL A           ; A CONTAINS 8*EXP.
166 F1E9 96CEF8                 ADD A, X        ; A CONTAINS DIGIT + 10*EXP.
167 F1EC AFC8                   PUSH A          ; SAVE BACK ON STACK.
168 F1EE C0                     LDS A, M(B+)
169 F1EF 40                     NOP             ; GET NEXT CHAR.
170 F1F0 9C00                   IFEQ A, 0       ; IS IT A NULL ?
171 F1F2 41                     JP $A10EX       ; YES SO ADD EXPLICIT AND IMPLICIT
172                                             ; 10'S EXPONENT.
173                                             ; GET HERE MEANS IT IS A DIGIT,
174 F1F3 7F                     JP $EXCLP       ; SO GO BACK AND ACCUMULATE IT.
175                     ;
176                     $A10EX:
177                     ; DONE COLLECTING DIGITS. ADD THE EXPLICIT AND IMPLICIT
178                     ; 10'S EXPONENT COLLECTED SO FAR.
179 F1F4 3FC8                   POP A           ; GET EXPLICIT 10'S EXPONENT.
180 F1F6 3FCA                   POP K           ; GET ITS SIGN.
181 F1F8 8200CAFC               IFEQ K, 0       ; IS IT POSITIVE ?
182 F1FC 42                     JP $ADDEX       ; YES SO ADD 'EM.
183 F1FD 01                     COMP A
184 F1FE 04                     INC A           ; CHANGE TO 2'S COM.
185                     $ADDEX:
186 F1FF AGFFFAC4F8             ADD A, W(SP-06) ; ADD IMPLICIT EXPONENT.
187 F204 BD7FFF                 IFGT A, 07FFF   ; IS IT NEGATIVE ?
188 F207 43                     JP $NEG10       ; YES, CHANGE IT.
189 F208 9300                   LD X, 0         ; LOAD POST. SIGN IN X.
190 F20A 44                     JP $ESAVE
191                     $NEG10
192 F20B 93FF                   LD, X, OFF      ; LOAD NEG. SIGN IN X.
193 F20D 01                     COMP A
194 F20E 04                     INC A           ; MAKE IT POSITIVE.
195                     $ESAVE:
196 F20F ACC8CC                 LD B, A         ; SAVE 10'S EXPONENT IN A.
197 F212 3FC8                   POP A           ; GET HI-INT.
198 F214 3FCA                   POP K           ; GET LO-INT.
199 F216 AFCE                   PUSH X          ; SAVE SIGN OF 10'S EXPONENT.
200 F218 AFCC                   PUSH B          ; AND ITS VALUE.
201                     ;
202                     ; NOW CONVERT HI-INT.LO-INT TO A NORMALIZED FLOATING POINT
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    19
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

                        ; NUMBER. THE BINARY EXPONENT IS COLLECTED IN B.
                        ;
F21A 9000                       IFGT A, 0       ; IF HI-INT IS NOT 0,
F21C 58                         JP $NORM2       ; NEED TO SHIFT K AND A.
F21D 8200CAFD                   IFGT K, 0       ; IF HI-INT IS 0, BUT NOT LO-INT,
F221 4E                         JP $NORM1       ; NEED TO SHIFT ONLY K.
                                                ; GET HERE MEANS MANTISSA IS 0.
F222 00                         CLR A
F223 ACC8CA                     LD K, A
F226 02                         SET C
F227 8208C4EB                   SUB SP, 08      ; ADJUST SP. DONE!!!
F22B 3FCC                       POP B
F22D 3FCE                       POP X
F22F 3C                         RET
                        ;
                        $NORM1:
                        ; HI-INT IS 0, SO WORK WITH LO-INT ONLY.
F230 AECA                       X A, K
F232 9210                       LD B, 010       ; LOAD 16 INTO EXPONENT COUNTER.
F234 42                         JP $NRLUP
                        ;
                        $NORM2:
                        ; HI-INT IS NOT 0, SO NEED TO HANDLE BOTH.
F235 9220                       LD B, 020       ; LOAD 32 INTO LOOP COUNTER.
                        $NRLUP:
F237 E7                         SHL A
F238 07                         IF C            ; DID A 1 COME OUT ?
F239 4D                         JP $NRDUN       ; YES IT IS NORMALIZED NOW.
F23A AECA                       X A, K
F23C E7                         SHL A
F23D 07                         IF C
F23E 96CA08                     SET K.0
F241 AECA                       X A, K
F243 AACC                       DECSZ B
F245 40                         NOP             ; SHOULD NEVER BE SKIPPED!!
F246 6F                         JP $NRLUP
                        $NRDUN:
F247 D7                         RRC A           ; RESTORE SHIFTED 1.
F248 AB00                       ST A, TMP1      ; STORE IN W(0).
F24A 3FCE                       POP X           ; GET 10'S EXPONENT.
F24C 3FC8                       POP A           ; GET 10'S EXPONENT SIGN.
F24E AE00                       X A, TMP1       ; A IS HI-INT ONCE MORE.
F250 AFCC                       PUSH B          ; SAVE BINARY EXPONENT.
F252 AFCE                       PUSH X          ; SAVE 10'S EXPONENT.
F254 AECA                       X A, K          ; HI-INT TO K, LO-INT TO A.
F256 960010                     IF TMP1.0       ; IS 10'S EXPONENT NEGATIVE ?
F259 58                         JP $DIV10       ; YES, GO TO DIVIDE BY 10.
                        ; GET HERE MEANS 10'S EXPONENT IS POSITIVE, SO MULTIPLY BY 10.
                        ; ACTUALLY, WHAT IS USED IS
                        ;       10^N = (0.625*(2^4)^N
                        ; SO MULTIPLY BY 0.625 NOW AND TAKE CARE OF 2^(4*N) LATER.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    20
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

254 F25A A4F26ACCAB             LD B, W($MTLO)
255 F25F AFCC                   PUSH B          ; SAVE LO WORD OF 0.625 ON STACK.
256 F261 A4F26CCCAB             LD B, W($MTHI)
257 F266 AFCC                   PUSH B          ; SAVE HI WORD OF 0.625 ON STACK.
258 F268 57                     JP $JAMIT       ; GO TO ROUTINE THAT JAMS 0.625^N
259                                             ; BY REPEATED MULTIPLICATION INTO HI-INT.LO-INT.
260                     ;
261                     ; DEFINE SOME CONSTANTS.
262 F269 40                             . EVEN  ; FORCE EVEN ADDRESS.
263 F26A 0000           $MTLO:  .WORD 0
264 F26C 00A0           $MTHI:  .WORD 0A000
265 F26E CDCC           $DTLO:  .WORD 0CCCD
266 F270 CCCC           $DTHI:  .WORD 0CCCC
267                     ;
268                     $DIV10:
269                     ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO DIVIDE BY 10.
270                     ; ACTUALLY WHAT IS DONE IS
271                     ;       10^(-N) = ((2^3)/(0.8))^(-N) = ((0.8)^N)*(2^(-3*N)))
272                     ; SO MULTIPLY BY 0.8 NOW AND TAKE CARE OF 2^(-3*N) LATER.
273 F272 A4F26ECCAB             LD B, W($DTLO)
274 F277 AFCC                   PUSH B          ; SAVE LO WORD OF .8
275 F279 A4F270CCAB             LD B, W($DTHI)
276 F27E AFCC                   PUSH B          ; SAVE HI WORD OF .8
277                     ;
278                     $JAMIT:
279                     ; JAM IN THE MULTIPLICATION PART NEEDED TO HANDLE THE 10'S EXP.
280 F280 9200                   LD B, 0         ; B IS USED TO TRACK ANY BINARY POWERS
281                                             ; THAT COME UP DURING NORMALIZATION.
282 F282 8200CEFC               IFEQ X, 0       ; IS 10'S EXPONENT 0 ?
283 F286 57                     JP $JAMDN       ; YES, DONE ALREADY.
284                     $JAMLP:
285 F287 35C0                   JSR BFMUL       ; MULTIPLY USING 32 BIT UNSIGNED.
286 F289 AECA                   X A, K          ; SWAP HI AND LO WORDS.
287 F28B E7                     SHL A
288 F28C 07                     IF C            ; IS THERE A CARRY ?
289 F28D 4A                     JP $ISNED       ; YES, SO IT IS ALREADY NORMALIZED.
290 F28E A9CC                   INC B           ; NEED TO SHIFT LEFT TO NORMALIZE, SO
291                                             ; INCREASE B BY 1.
292 F290 AECA                   X A, K
293 F292 E7                     SHL A
294 F293 07                     IS C
295 F294 96CA08                 SET K.0
296 F297 43                     JP $OVR1
297                     $ISNED:
298 F298 D7                     RRC A
299 F299 AECA                   X A, K
300                     $OVR1:
301 F29B AACE                   DECSZ X         ; DONE YET ?
302 F29D 76                     JP $JAMLP       ; NO SO DO IT ONCE MORE.
303                     ; GET HERE MEANS MULTIPLICATIONS HAVE BEEN DONE. NOW TAKE
304                     ; CARE OF THE EXPONENTS.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    21
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

305                     $JAMDN:
306 F29E 3FCE                   POP X
307 F2A0 3FCE                   POP X           ; GET 0.625 OR 0.8 OFF THE STACK.
308 F2A2 3FCE                   POP X           ; GET THE 10'S EXPONENT.
309 F2A4 AFC8                   PUSH A          ; SAVE LO WORD OF FLP NUMBER.
310 F2A6 AFCA                   PUSH K          ; SAVE HI WORD OF FLP NUMBER.
311 F2A8 A6FFFAC4A8             LD A, W(SP-6)   ; GET THE BINARY EXPONENT THAT WAS SAVED.
312 F2AD 02                     SET C
313 F2AE 96CCEB                 SUBC A, B       ; SUBTRACT FROM IT BINARY EXPONENT COLLECTED
314                                             ; DURING THE JAMMING.
315 F2B1 ACC8CC                 LD B, A         ; SAVE IT IN B.
316 F2B4 A8CE                   LD A, X         ; MOVE THE 10'S EXPONENT TO A.
317 F2B6 960010                 IF TMP1.0       ; IS THE 10'S EXPONENT NEGATIVE ?
318 F2B9 49                     JP $NAGAS       ; YES, SO GOT TO SUBTRACT.
319                                             ; GET HERE MEANS 10'S EXPONENT IS
320                                             ; POSITIVE, SO MUL IT BY 4.
321 F2BA E7                     SHL A           ; MULTIPLY BY 2.
322 F2BB E7                     SHL A           ; MULTIPLY BY 2 AGAIN.
323 F2BC 96CCF8                 ADD A, B        ; GET THE BINARY EXPONENT IN ALSO.
324 F2BF B8007E                 ADD A, 07E      ; AND THE IEEE BIAS.
325 F2C2 4C                     JP $EXCPT       ; GO CHECK FOR OVER/UNDERFLOW.
326                     ;
327                     $NAGAS:
328                     ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO GOT TO MULTIPLY
329                     ; IT BY -3.
330 F2C3 E7                     SHL A           ; MULTIPLY BY 2.
331 F2C4 96CEF8                 ADD A, X        ; ADD TO GIVE MULTIPLY BY 3.
332 F2C7 01                     COMP A
333 F2C8 04                     INC A           ; MAKE IT NEGATIVE.
334 F2C9 96CCF8                 ADD A, B        ; GET IN THE BINARY EXPONENT.
335 F2CC B8007E                 ADD A, 07E      ; AND THE IEEE BIAS.
336                     $EXCPT:
337                     ; CHECK FOR OVERFLOW/UNDERFLOW.
338 F2CF ACC4CE                 LD X, SP        ; FIRST DO SOME JUGGLING
339 F2D2 02                     SET C           ; TO BE COMPATIBLE WITH EXCEPTION
340 F2D3 820ACEEB               SUBC X, 0A      ; HANDLING IN OTHER ROUTINES.
341 F2D7 AFCE                   PUSH X
342 F2D9 BD7FFF                 IFGT A, 07FFF   ; IS BIASED EXPONENT NEGATIVE ?
343 F2DC B4FD3F                 JMPL UNDFL
344 F2DF 9C00                   IFEQ A, 0       ; IS IT 0 ?
345 F2E1 B4FD3A                 JMPL UNDFL      ; YES IT IS STILL UNDERFLOW.
346 F2E4 9DFE                   IFGT A, 0FE     ; IS IT GT THAN 254 ?
347 F2E6 B4FD46                 JMPL OVRFL
348                     ; GET HERE MEANS VALID SP FLP NUMBER.
349 F2E9 3FCE                   POP X           ; X POINTS TO MANTISSA SIGN.
350 F2EB E7                     SHL A
351 F2EC E7                     SHL A
352 F2ED E7                     SHL A
353 F2EE E7                     SHL A
354 F2EF E7                     SHL A
355 F2FD E7                     SHL A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    22
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

356 F2F1 E7                     SHL A
357 F2F2 E7                     SHL A           ; MOVE EXPONENT TO HIGH BYTE.
358 F2F3 8FFA                   OR A, W(X)      ; GET THE MANTISSA SIGN IN.
359 F2F5 AB00                   ST A, TMP1      ; SAVE IT IN TMP1.
360 F2F7 3FCA                   POP K           ; F1-HI TO K.
361 F2F9 3FC8                   POP A           ; F1-LO TO A.
362 F2FB F1                     X A, W(X+)      ; SAVE F1-LO.
363 F2FC A8CA                   LD A, K
364 F2FE F1                     X A, W(X+)      ; SAVE F1-HI.
365 F2FF A800                   LD A, TMP1
366 F301 F3                     X A, W(X-)      ; SAVE F1-EXP.F1-SIGN, X POINTS TO F1-HI.
367 F302 3664                   JSRL SROUND     ; ROUND THE RESULT.
368 F304 F2                     LD A, W(X-)     ; X POINTS TO F1-HI.
369 F305 F2                     LD A, W(X-)     ; X POINTS TO F1-LO.
370 F306 AFCE                   PUSH X
371 F308 368C                   JSR FPAK        ; PACK IT INTO IEEE FORMAT.
372 F30A 3FC4                   POP SP
373 F30C 3FCC                   POP B
374 F30E 3FCE                   POP X
375 F310 3C                     RET
376                         ;
377                                 .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    23
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
FTOA.MAC

                                    .FORM 'FTOA.MAC'
                                    .INCLD FTOA.MAC
                                    .TITLE FTOA
                                    .LOCAL
                        ;
                        ; THIS SUBROUTINE CONVERTS A SINGLE PRECISION, BINARY FLOATING
                        ; POINT NUMBER IN THE IEEE FORMAT TO A DECIMAL FLOATING POINT
                        ; STRING. THE DECIMAL FLOATING POINT STRING IS OBTAINED TO A
                        ; PRECISION OF 9 DECIMAL DIGITS.
                        ;
                        ; THE ALGORITHM USED IS BASED ON:
                        ; J.T. COONEN, 'AN IMPLEMENTATION GUIDE TO A PROPOSED STANDARD
                        ; FOR FLOATING POINT ARITHMETIC,' IEEE COMPUTER, JAN. 1980, PP 68-79.
                        ;
                        ; ON INPUT, THE BINARY SP FLP NUMBER IS IN REGS. K AND A.
                        ; B CONTAINS THE ADDRESS OF THE LOCATION WHERE THE DECIMAL FLOATING
                        ; POINT STRING IS TO START. NOTE THAT AT LEAST 17 BYTES ARE NEEDED
                        ; FOR THE STORAGE OF THE STRING. THE LAST BYTE IS ALWAYS NULL.
                        ;
                        ; ALL REGISTERS ARE PRESERVED BY THIS SUBROUTINE.
                        ;
                        FTOA:
F311 AFCE                       PUSH X          ; SAVE X ON THE STACK.
F313 AFCC                       PUSH B          ; SAVE B ON THE STACK.
                        ; CHECK AND SEE IF F1 IS A NAN.
F315 3605                       JSR FNACHK
F317 07                         IF C
F318 B401B4                     JMPL $NAN       ; YET IT IS, SO GET OUT.
                        ; CHECK AND SEE IF F1 IS ZERO.
F31B 36CA                       JSR FZCHK
F31D 07                         IF C
F31E B401C8                     JMPL $ZERO      ;YES IT IS, SO GET OUT.
                        ; GET HERE MEANS F1 IS A NON-ZERO, NON-NAN FLP NUMBER.
F321 ACC4CE                     LD X, SP
F324 8206C4F8                   ADD SP, 06      ; ADJUST SP.
F328 36C7                       JSR FUNPAK      ; UNPACK THE NUMBER.
                                                ; X POINTS ONE WORD PAST F1-EXP.F1-SIGN
                                                ; ON RETURN.
                        ;
                        ; COMPUTE THE EXPONENT OF 10 FOR DECIMAL FLP NO.
                        ; THIS IS DONE AS FOLLOWS:
                        ;       SUPPOSE F1 = FM * (2^M)
                        ;       LET U = M*LOG(2)        NOTE: LOG IS TO BASE 10.
                        ;       THEN V = INT(U+1-9)
                        ;       IS USED AS THE 10'S EXPONENT.
                        ;                               NOTE: INT REFERS TO INTEGER PART.
                        ;
F32A D2                         LD A, M(X-)     ; X POINTS TO F1-EXP.
F32B D2                         LD A, M(X-)     ; LOAD F1-EXP. X POINTS TO F1-SIGN.
F32C B7000000                   LD TMP1, 0      ; FIRST GUESS POSITIVE SIGN FOR EXP.
F330 B8FF82                     ADD A, 0FF82    ; REMOVE IEEE BIAS FROM F1-EXP.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    24
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

  50 F333 AFC8                  PUSH A          ; SAVE IT ON THE STACK.
  51 F335 AFCE                  PUSH X          ; SAVE F1-SIGN ADDRESS ALSO.
  52 F337 07                    IF C            ; WAS THERE A CARRY ON THE LAST ADD ?
  53 F338 46                    JP $MLOG2       ; YES, SO 2'S EXP IS POSITIVE.
  54 F339 B700FF00              LD TMP1, OFF    ; 2'S EXPONENT IS NEGATIVE.
  55 F33D 01                    COMP A
  56 F33E 04                    INC A           ; MAKE IT POSITIVE.
  57                        $MLOG2:
  58                        ; MULTIPLY M BY LOG(2).
  59 F33F BE4D10                MULT A, 04D10   ; LOG(2) IS 0.0100110100010000 TO 16 BITS.
  60                                            ; X CONTAINS INTEGER PART, AND A FRACT. PART.
  61 F342 AECE                  X A, X          ; SWAP THE TWO.
  62 F344 960010                IF TMP1.0       ; WAS THE 2'S EXPONENT NEGATIVE ?
  63 F347 41                    JP $CSIGN       ; YES, SO MAKE U NEGATIVE.
  64 F348 4B                    JP $REMV9       ; NO, SO GO DO V = U + 1 - 9.
  65                        $CSIGN:
  66 F349 01                    COMP A          ; COMP INTEGER PART.
  67 F34A AECE                  X A, X
  68 F34C 01                    COMP A          ; FRACTION PART.
  69 F34D B80001                ADD A, 01
  70 F350 AECE                  X A, X
  71 F352 07                    IF C
  72 F353 04                    INC A
  73                        $REMV9:
  74 F354 04                    INC A           ; INCREASE FRACTION PART.
  75 F355 B8FFF7                ADD A, 0FFF7    ; SUBTRACT 9.
  76 F358 BD7FFF                IFGT A, 07FFF   ; IS IT NEGATIVE ?
  77 F35B 45                    JP $CHNGS       ; YES, SO CHANGE ITS SIGN.
  78 F35C B7000000              LD TMP1, 0      ; REMEMBER POSITIVE SIGN.
  79 F36D 4F                    JP $DIV10
  80                        $CHNGS:
  81 F361 B700FF00              LD TMP1, OFF    ; REMEMBER NEGATIVE SIGN.
  82 F365 01                    COMP A          ; MAKE V POSITIVE.
  83 F366 AECE                  X A, X
  84 F368 01                    COMP A
  85 F369 B80001                ADD A, 01
  86 F36C AECE                  X A, X
  87 F36E 07                    IF C
  88 F36F 04                    INC A
  89                        $DIV10:
  90                        ;
  91                        ; V = INT (U+1-9) HAS BEEN COMPUTED AND IS IN A.
  92                        ; NOW COMPUTE W = F1/(10^V). W SHOULD BE AN INTEGER, AND IT IS
  93                        ; COMPUTED TO A 32 BIT PRECISION.
  94                        ; THIS COMPUTATION IS DONE AS FOLLOWS:
  95                        ;       IF V > 0, THEN F1/(10^V) = F1*(0.8^V)*(2^(-3V)).
  96                        ;       IF V < 0, THEN F1/(10^V) = F1*(0.625^U)*(2^(4V)).
  97                        ; SO FIRST MULTIPLY THE MANTISSA OF F1 V TIMES BY 0.8 (OR 0.625)
  98                        ; AND THEN ADJUST THE EXPONENT OF F1. NOTE THAT THE PARTIAL PRODUCTS
  99                        ; IN MULTIPLYING BY 0.8 (OR 0.625) ARE KEPT NORMALIZED. THIS IS
 100                        ; ESSENTIAL TO PRESERVE 32 BIT ACCURACY IN THE FINAL RESULT.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    25
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

                        ; SINCE THE MANTISSA OF F1 IS NORMALIZED, AND 0.8 (OR 0.625 IS ALSO
                        ; NORMALIZED, EACH PRODUCT NEEDS AT MOST 1 LEFT SHIFT FOR
                        ; RENORMALIZATION. THE SHIFTS ACCUMULATED DURING RENORMALIZATION ARE
                        ; TRACKED AND ACCOUNTED FOR IN THE CALCULATION.
F370 3FCE                       POP X           ; X NOW POINTS TO F1-SIGN.
F372 AFC8                       PUSH A          ; SAVE U ON THE STACK.
F374 ACC8CC                     LD B, A         ; MOVE V TO B ALSO.
F377 F2                         LD A, W(X-)     ; X POINTS TO F1-HI.
F378 F2                         LD A, W(X-)     ; LOAD F1-HI. X POINTS TO F1-LO.
F379 ACC8CA                     LD K, A
F37C F4                         LD A, W(X)      ; LOAD F1-LO.
F370 960010                     IF TMP1.0       ; IS V NEGATIVE?
F380 57                         JP $MUL10       ; YES, SO MULTIPLY V TIMES BY .625.
                                                ; GET HERE MEANS MULTIPLY V TIMES BY .8.
F381 A4F390CEAB                 LD X, W($DTLO)
F386 AFCE                       PUSH X          ; LO WORD OF 0.8 TO STACK.
F388 A4F392CEAB                 LD X, W($DTHI)
F38D AFCE                       PUSH X          ; HI WORD OF 0.8 TO STACK.
F38F 56                         JP $JAMIT       ; GO DO MULTIPLICATION.
                        ;
                                .EVEN           ; FORCE EVEN ADDRESS.
F390 CDCC               $DTLO:  .WORD OCCCD
F392 CCCC               $DTHI:  .WORD OCCCC
F394 0000               $MILO:  .WORD 0
F396 00A0               $MTHI:  .WORD 0A000
                        ;
                        $MUL10:
F398 A4F394CEAB                 LD X, W($MTLO)
F39D AFCE                       PUSH X          ; LO WORD OF 0.625 TO STACK.
F39F A4F396CEAB                 LD X, W($MTHI)
F3A4 AFCE                       PUSH X          ; HI WORD OF 0.625 TO STACK.
                        ;
                        $JAMIT:
F3A6 9300                       LD X, 0         ; INIT X TO TRACK ANY POWERS OF
                                                ; 2 GENERATED DURING NORMALIZATION
                                                ; OF PARTIAL PRODUCTS.
F3A8 8200CCFC                   IFEQ B, 0       ; IS B ALREADY 0 ?
F3AC 57                         JP $JAMON       ; YES, SO SKIP MULTIPLY LOOP.
                        $JAMLP:
F3AD 36E6                       JSR BFMUL       ; MULTIPLY.
F3AF AECA                       X A, K          ; SWAP HI AND LO WORDS OF PART. PROD.
F3B1 E7                         SHL A
F3B2 07                         IF C            ; IS THERE A CARRY ?
F3B3 4A                         JP $ISNED       ; YES, SO SKIP OVER RENORMALIZATION.
                                                ; GET HERE MEANS NEED TO RENORM.
F3B4 A9CE                       INC X           ; UPDATE RENORM COUNT.
F3B6 AECA                       X A, K          ; SWAP HI AND LO PART. PROD.
F3B8 E7                         SHL A
F3B9 07                         IF C
F3BA 96CA08                     SET K.0         ; SET BIT SHIFTED OUT FROM LO WORD.
F3BD 43                         JP $OVR1
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    26
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

152                         $ISNED:
153 F3BE D7                         RRC A           ; PUT BACK SHIFTED BIT.
154 F3BF AECA                       X A, K
155                         $OVR1:
156 F3C1 AACC                       DECSZ B         ; IS B 0 YET ?
157 F3C3 76                         JP $JAMLP       ; NO, SO DO IT AGAIN.
158                         $JAMON:
159                         ; GET HERE MEANS MULTIPLICATION HAS BEEN DONE, SO TAKE CARE
160                         ; OF EXPONENT.
161 F3C4 3FCC                       POP B
162 F3C6 3FCC                       POP B           ; GET RID OF 0.8 (OR 0.625) FROM STACK.
163 F3C8 AFC8                       PUSH A          ; SAVE LO WORD OF PROD.
164 F3CA AFCA                       PUSH K          ; SAVE HI WORD OF PRODUCT.
165 F3CC A6FFF8C4A8                 LD A, W(SP-08)  ; GET F1'S BINARY EXPONENT.
166 F3D1 02                         SET C
167 F3D2 96CEEB                     SUBC A, X       ; SUBTRACT FROM IT ANYTHING COLLECTED
168                                                 ; DURING RENORM.
169 F3D5 ACC8CE                     LD X, A         ; AND SAVE IT IN X.
170 F3D8 A6FFFAC4A8                 LD A, W(SP-06)  ; GET V FROM THE STACK.
171 F3DD 960010                     IF TMP1.0       ; IS V NEGATIVE ?
172 F3E0 49                         JP $ML4         ; YES, SO MULTIPLY V BY 4.
173                                                 ; GET HERE MEANS MULTIPLY V BY -3.
174 F3E1 E7                         SHL A           ; NOW A CONTAINS 2*V.
175 F3E2 A6FFFAC4F8                 ADD A, W(SP-06) ; NOW A CONTAINS 3*V.
176 F3E7 01                         COMP A
177 F3E8 04                         INC A           ; NOW A CONTAINS -3*V.
178 F3E9 42                         JP $ADEM        ; GO FIGURE FINAL EXPONENT.
179                         $ML4:
180 F3EA E7                         SHL A
181 F3EB E7                         SHL A           ; NOW A CONTAINS 4*V.
182                         $ADEM:                  ;
183 F3EC 96CEF8                     ADD A, X        ; A SHOULD NOW BE A POSITIVE INTEGER
184                                                 ; IN THE RANGE 0 TO 32.
185                         ; NOW CHECK AND SEE IF A HAS ENOUGH PRECISION.
186 F3EF 9020                       IFGT A, 020     ; NEED MORE THAN 32 BITS ?
187 F3F1 5A                         JP $INCRV       ; YES, SO GO INCREASE V.
188 F3F2 9D1B                       IFGT A, 01B     ; NEED AT LEAST 28 BITS ?
189 F3F4 9435                       JMP $GOON       ; YES, SO ALL IS OK. GO ON.
190                                                 ; GET HERE MEANS NEED MORE
191                                                 ; PRECISION, SO DECREASE V.
192 F3F6 3FC8                       POP A           ; GET HI-PROD OFF STACK.
193 F3F8 3FC8                       POP A           ; GET LO WORD OFF STACK.
194 F3FA 3FC8                       POP A           ; GET MAGN. OF V.
195 F3FC 96D010                     IF TMP1.0       ; IS V NEG. ?
196 F3FF 56                         JP $VUP         ; YES, SO GO INCR. MAGN. OF V.
197                                                 ; GET HERE MEANS V IS POSITIVE,
198                                                 ; AND NEED TO DECREMENT IT.
199 F400 B8FFFF                     ADD A, 0FFFF    ; SUBTRACT 1 FROM A.
200 F403 07                         IF C            ; GOT A CARRY ?
201 F404 5A                         JP $GOBAK       ; THEN OK.
202 F405 01                         COMP A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    27
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

203 F406 04                     INC A
204 F407 B700FF00               LD TMP1, 0FF     ; U CHANGES SIGN.
205 F40B 53                     JP $GOBAK
206                     $INCRV:
207 F40C 3FC8                   POP A            ; GET HI PROD. OFF STACK.
208 F40E 3FC8                   POP A            ; GET LO PROD. OFF STACK.
209 F410 3FC8                   POP A            ; GET MAGN. OF V.
210 F412 960010                 IF TMP1.0        ; IS V NEGATIVE ?
211 F415 42                     JP $VDOWN        ; YES.
212                     $VUP:
213 F416 04                     INC A
214 F417 47                     JP $GOBAK
215                     $VDOWN:
216 F418 AAC8                   DECSZ A
217 F41A 44                     JP $GOBAK
218 F41B B7000000               LD TMP1, 0       ; V CHANGES SIGN.
219                     $GOBAK:
220 F41F ACC4CE                 LD X, SP
221 F422 02                     SET C
222 F423 8204CEEB               SUBC X, 04
223 F427 AFCE                   PUSH X
224 F429 95B9                   JMP $DIV10
225                     $GOON:
226 F42B 01                     COMP A
227 F42C 04                     INC A            ; NEGATE A.
228 F42D B80020                 ADD A, 020       ; SUBTRACT IT FROM 32.
229 F430 ACC8CE                 LD X, A          ; AND MOVE IT TO X.
230 F433 3FCA                   POP X            ; GET HI WORD OF PRODUCT.
231 F435 3FC8                   POP A            ; GET LO WORD OF PROD.
232 F437 8200CEFC               IFEQ X, 0        ; IS X 0 ?
233 F43B 49                     JP $INDUN        ; YES, SO ALREADY A 32 BIT INTEGER.
234                     $INTFY:
235                     ; NOW ADJUST THE PRODUCT TO FORM A 32 BIT INTEGER.
236 F43C AECA                   X A, K           ; SWAP HI AND LO WORDS.
237 F43E C7                     SHR A
238 F43F AECA                   X A, K
239 F441 D7                     RRC A            ; SHIFT IT RIGHT ONCE.
240 F442 AACE                   DECSZ X          ; X 0 YET ?
241 F444 68                     JP $INTFY        ; NO SO GO DO SOME MORE.
242                     $INDUN:
243                     ; GET HERE MEANS K.A CONTAIN THE 32 BIT INTEGER THAT IS THE
244                     ; MANTISSA OF THE DECIMAL FLP NUMBER.
245 F445 AFC8                   PUSH A           ; SAVE LO-INT.
246 F447 AFCA                   PUSH K           ; SAVE HI INT.
247 F449 A6FFF0C4A8             LD A, W(SP-010)  ; GET STARTING ADDRESS OF DECIMAL STRING.
248 F44E B80010                 ADD A, 010       ; ADD 16 TO IT.
249 F451 ACC8CC                 LD B, A          ; AND MOVE IT B.
250 F454 00                     CLR A
251 F455 C3                     XS A, M(B-)      ; OUTPUT TERMINATING NULL BYTE.
252 F456 40                     NOP
253 F457 A6FFFAC4A8             LD A, W(SP-06)   ; GET V.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    28
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

254 F45C 9F0A                   DIV A, 0A       ; DIVIDE IT BY 10. QUOT. IN A,
255                                             ; REM. IN X.
256 F45E AECE                   X A, X          ; REM TO A.
257 F460 B80030                 ADD A, 030      ; MAKE IT INTO ASCII BYTE.
258 F463 C3                     XS A, M(B-)     ; OUTPUT IT.
259 F464 40                     NOP
260 F465 A8CE                   LD A, X
261 F467 B80030                 ADD A, 030
262 F46A C3                     XS A, M(B-)
263 F46B 40                     NOP             ; FINISHED OUTPUTING EXPONENT.
264 F46C 902B                   LD A, 028       ; SAY EXP SIGN IS '+'.
265 F46E 960010                 IF TMP1.0
266 F471 902D                   LD A, 02D       ; NOPE, IT IS '-'.
267 F473 C3                     XS A, M(B-)     ; OUTPUT IT.
268 F474 40                     NOP
269 F475 9045                   LD A, 045
270 F477 C3                     XS A, M(B-)     ; OUTPUT 'E'.
271 F478 40                     NOP
272 F479 902E                   LD A, 02E
273 F47B C3                     XS A, M(B-)     ; OUTPUT '.'.
274 F47C 40                     NOP
275                         ; NOW NEED TO OUTPUT 10 DECIMAL DIGITS.
276 F47D B7000A00               LD TMP1, 0A     ; LOAD 10 INTO TMP1 AS LOOP COUNTER.
277                         $DOLUP:
278 F481 3FC8                   POP A           ; A CONTAINS HI INT.
279 F483 9F0A                   DIV A, 0A       ; DIVIDE IT BY 10. QUOT. IN A,
280                                             ; REM. IN X.
281 F485 ACC8CA                 LD K, A
282 F488 3FC8                   POP A           ; A CONTAINS LO INT.
283 F48A AFCC                   PUSH B          ; SAVE DEC. STR. ADDR.
284 F48C ACCACC                 LD B, K         ; B CONTAINS HI-QUOT.
285 F48F 82                     .BYTE 082,0A,0C8,0EF
    F490 0A
    F491 C8
    F492 EF
286                         ; THE ABOVE 4 BYTES REPRESENT THE INSTRUCTION DIVD A, 0A.
287                         ; BECAUSE THE ASSEMBLER DOES NOT KNOW ABOUT IT YET, WE HAVE TO
288                         ; KLUDGE IT THIS WAY.
289                         ; AFTER THE DIVD, A CONTAINS THE LO-QUOT. AND X THE REM.
290 F493 ACCCCA                 LD K, B         ; MOVE HI-QUOT TO K.
291 F496 3FCC                   POP B           ; B CONTAINS DEC. STR. ADDR.
292 F498 AFC8                   PUSH A          ; SAVE LO INT.
293 F49A AFCA                   PUSH K          ; SAVE HI INT.
294 F49C A8CE                   LD A, X         ; MOVE REM TO A.
295 F49E B80030                 ADD A, 030      ; ASCII-FY IT.
296 F4A1 C3                     XS A, M(B-)     ; AND OUTPUT IT.
297 F4A2 40                     NOP
298 F4A3 AA00                   DECSZ TMP1      ; IS TMP1 0 YET ?
299 F4A5 9524                   JMP $DOLUP      ; NO, GO GET SOME MORE.
300                         ; GET HERE MEANS DONE WITH OUTPUTING MANTISSA.
301 F4A7 3FC8                   POP A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    29
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

302 F4A9 3FC8                    POP A
303 F4AB 3FC8                    POP A
304 F4AD 3FC8                    POP A           ; GET SOME GARBAGE OFF THE STACK.
305 F4AF 3FCA                    POP K           ; GET F1-EXP.F1-SIGN TO K.
306 F4B1 9020                    LD A, 02D       ; LOAD SP INTO A.
307 F4B3 96CA10                  IF K.0
308 F4B6 902D                    LD A, 02D       ; IF MAINT. IS NEG. LOAD '-'.
309 F4B8 C6                      ST A, M(B)      ; OUTPUT SIGN.
310 F4B9 AFCA                    PUSH K          ; F1-EXP.F1-SIGN BACK ON STACK.
311 F4BB ACC4CE                  LD X, SP
312 F4BE 02                      SET C
313 F4BF 8206CEEB                SUBC X, 06      ; X POINTS TO F1-LO.
314 F4C3 AFCE                    PUSH X
315 F4C5 B5FBB4     +            JSR FPAK        ; PACK IT, SO RESTORING K AND A.
316 F4C8 3FC4                    POP SP
317 F4CA 3FCC                    POP B           ; RESTORE B.
318 F4CC 3FCE                    POP X           ; RESTORE X.
319 F4CE 3C                      RET
320                         ;
321                         $NAN:
322                         ; GET HERE MEANS F1 IS A NAN.
323 F4CF AFC8                    PUSH A
324 F4D1 ACCCCE                  LD X, B
325 F4D4 8210CEF8                ADD X, 010
326 F4D8 00                      CLR A
327 F4D9 03                      X A, M(X-)
328 F4DA 9210                    LD B, 010
329                         $NANLP:
330 F4DC 90FF                    LD A, 0FF
331 F4DE D3                      X A, M(X-)
332 F4DF AACC                    DECSZ B
333 F4E1 65                      JP $NANLP
334 F4E2 3FC8                    POP A
335 F4E4 3FCC                    POP B
336 F4E6 3FCE                    POP X
337 F4E8 3C                      RET
338                         ;
339                         $ZERO:
340                         ; GET HERE MEANS F1 IS ZERO.
341 F4E9 AFC8                    PUSH A
342 F4EB ACCCCE                  LD X, B         ; X CONTAINS DECIMAL STRING ADDR.
343 F4EE 8210CEF8                ADD X, 010
344 F4F2 00                      CLR A
345 F4F3 D3                      X A, M(X-)      ; OUTPUT TERMINATING NULL BYTE.
346 F4F4 9030                    LD A, 030       ; LOAD 0 INTO A.
347 F4F6 D3                      X A, M(X-)
348 F4F7 9030                    LD A, 030
349 F4F9 D3                      X A, M(X-)      ; OUTPUT 00 FOR EXPONENT.
350 F4FA 902B                    LD A, 02B       ; LOAD '+' SIGN.
351 F4FC D3                      X A, M(X-)
352 F4FD 9045                    LD A, 045       ; LOAD 'E'
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    30
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

353 F4FF D3                         X A, M(X-)
354 F500 902E                       LD A, 02E       ; LOAD '.'.
355 F502 D3                         X A, M(X-)
356 F503 920A                       LD B, 0A
357                         $ZERLP:
358 F505 9030                       LD A, 030
359 F507 D3                         X A, M(X-)
360 F508 AACC                       DECSZ B
361 F50A 65                         JP $ZERLP
362 F508 9020                       LD A, 020       ; LOAD SP.
363 F50D D5                         X A, M(X)
364 F50E 3FC8                       POP A
365 F510 3FCC                       POP B
366 F512 3FCE                       POP X
367 F514 3C                         RET
368                         ;
369                                 .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    31
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FADD.MAC

                                    .FORM, 'FADD.MAX'
                                    .INCLD FADD.MAC
                                    .TITLE FADD
                                    .LOCAL
                    ;
                    ; SUBROUTINE TO ADD/SUBTRACT TWO SP FLOATING POINT NUMBERS.
                    ;          C = F1 + F2 OR C = F1 - F2
                    ;
                    ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                    ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                    ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                    ;
                    ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                    ; STACK POINTER ON ENTRY, THEN
                    ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                    ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                    ;
                    ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                    ;
                    FSUB:
F515 AB00                   ST A, TMP1
F517 A6FFFAC4A8             LD A, W(SP-06)   ; LOAD F2-R0.
F51C AFC8                   PUSH A           ; AND SAVE ON STACK.
F51E A6FFFAC4A8             LD A, W(SP-06)   ; LOAD F2-R1.
F523 BB8000                 XOR A, 08000     ; CHANGE THE SIGN.
F526 AFC8                   PUSH A           ; AND SAVE ON THE STACK.
F528 A800                   LD A, TMP1       ; RESTORE A.
F52A 3009                   JSR FADD         ; CALL THE ADD ROUTINE.
F52C AB00                   ST A, TMP1       ; SAVE A.
F52E 3FC8                   POP A            ; GET RID OF JUNK
F530 3FC8                   POP A            ; FROM THE STACK.
F532 A800                   LD A, TMP1       ; RESTORE A.
F534 3C                     RET
                    ;
                    FADD:
                    ; SAVE ADDRESS OF F2-R0 IN TMP1.
F535 AFCE                   PUSH X           ; SAVE X ON ENTRY.
F537 AFCC                   PUSH B           ; AND B ON ENTRY.
F539 ACC4CE                 LD X, SP
F53C 86FFF6CEF8             ADD X, 0FFF6     ; SUBTRACT 10.
F541 ACCE00                 LD TMP1, X       ; AND SAVE IN TMP1.
                    ; CHECK AND SEE IF F1 IS A NAN.
F544 B5FAF9      +          JSR FNACHK
F547 07                     IF C
F548 B4FAC4                 JMPL FNAN        ; F1 IS A NAN.
                    ; CHECK AND SEE IF F2 IS A NAN.
F54B ACCACC                 LD B, K
F54E ACC8CE                 LD X, A
F551 A20200A8               LD A, W(TMP1+2)
F555 ACC8CA                 LD K, A
F558 AECE                   X A, X
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    32
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

  50 F55A B5FAE3      +          JSR FNACHK
  51 F55D 07                     IF C
  52 F55E B4FAAE                 JMPL FNAN        ; F2 IS NAN.
  53                        ; CHECK AND SEE IF F2 IS ZERO.
  54 F561 B5FAED      +          JSR FZCHK
  55 F564 06                     IFN C
  56 F565 48                     JP $F1CHK        ; F2 IS NOT ZERO. CHECK F1.
  57 F566 ACCCCA                 LD K, B          ; F2 IS ZERO, SO ANSWER IS F1.
  58 F569 3FCC                   POP B
  59 F56B 3FCE                   POP X
  60 F56D 3C                     RET
  61                        ; CHECK AND SEE IF F1 IS ZERO.
  62                        $F1CHK:
  63 F56E ACCCCA                 LD K, B          ; RESTORE F1-R1 FROM B.
  64 F571 B5FADD      +          JSR FZCHK
  65 F574 06                     IFN C
  66 F575 4F                     JP $NTZERO       ; JUMP SINCE F1 IS ALSO NOT ZERO.
  67 F567 A20200AB               LD A, W(TMP1+2)  ; GET HERE MEANS F1 IS ZERO,
  68 F57A ACC8CA                 LD K, A          ; SO ANSWER IS F2.
  69 F57D AD00A8                 LD A, W(TMP1)
  70 F580 3FCC                   POP B
  71 F582 3FCE                   POP X
  72 F584 3C                     RET
  73                        ; GET HERE MEANS NORMAL ADDITION.
  74                        ; UNPACK F1 AND F2.
  75                        $NTZERO:
  76 F585 ACC4CE                 LD X, SP         ; X POINTS TO F1-LO.
  77 F588 8210C4F8               ADD SP, 010      ; MOVE SP PAST LOCAL STORAGE.
  78 F58C AFCE                   PUSH X           ; SAVE SP ON STACK FOR QUICK RETURN.
  79 F58E B5FAD0      +          JSR FUNPAK       ; UNPACK F1.
  80 F591 AC00CC                 LD B, TMP1       ; B NOW POINTS TO F2-R0.
  81 F594 ACCE00                 LD TMP1, X       ; TMP1 NOW POINTS TO F2-LO.
  82 F597 E0                     LDS A, W(B+)     ; LOAD F2-R0 INTO A.
  83 F598 40                     NOP
  84 F599 AECA                   X A, K
  85 F59B E4                     LD A, W(B)
  86 F59C AECA                   X A, K           ; LOAD F2-R1 INTO K.
  87 F59E B5FAC0      +          JSR FUNPAK       ; UNPACK F2.
  88                        ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
  89 F5A1 F2                     LD A, W(X-)
  90 F5A2 AC00CC                 LD B, TMP1
  91 F5A5 E2                     LDS A, W(B-)
  92 F5A6 40                     NOP
  93                        ; COMPARE F1-EXP AND F2-EXP.
  94 F5A7 F2                     LD A, W(X-)      ; LOAD F2-EXP.F2-SIGN INTO A.
  95 F5A8 B9FF00                 AND A, OFF00     ; MASK OUT SIGN.
  96 F5AB A6FFFCC4AB             ST A, W(SP-4)    ; SAVE IN C-SIGN.C-EXP.
  97 F5B0 E2                     LDS A, W(B-)
  98 F5B1 40                     NOP              ; LOAD F1-EXP.F1-SIGN INTO A.
  99 F5B2 B9FF00                 AND A, OFF00     ; CHANGE TO F1-EXP.00000000.
 100 F5B5 02                     SET C
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    33
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

101 F5B6 A6FFFCC4EB             SUBC A, W(SP-4) ; SUBTRACT F2-EXP.00000000.
102 F5BB 06                     IFN C
103 F5BC 942D                   JMP $F2GTR      ; F2-EXP IS BIGGER THAN F1-EXP.
104                     ; GET HERE MEANS F1-EXP IS BIGGER THAN F2-EXP.
105 F5BE 80C9CAAB               LD K, H(A)      ; SAVE DIFF. IN K TO BE USED AS LOOP COUNTER.
106 F5C2 A6FFFCC4FB             ADD A, W(SP-4)
107 F5C7 A6FFFCC4AB             ST A, W(SP-4)   ; RESTORE F1-EXP AND STORE IN C-SIGN.
108 F5CC 8217CAFD               IFGT K, 017
109 F5D0 51                     JP $ZROF2       ; K GT 23-DEC MEANS F2 GETS ZEROED IN SHIFTS.
110                     ; LOOP TO SHIFT F2 INTO ALIGNMENT.
111 F5D1 8200CAFC               IFEQ K, 0
112 F5D5 943B                   JMP $ADDMN      ; K = 0 MEANS DONE SHIFTING.
113                     $LOOP2:
114 F5D7 F4                     LD A, W(X)
115 F5D8 C7                     SHR A
116 F5D9 F3                     X A, W(X-)
117 F5DA F4                     LD A, W(X)
118 F5DB D7                     RRC A
119 F5DC F1                     X A, W(X+)
120 F5DD AACA                   DECSZ K
121 F5DF 68                     JP $LOOP2
122 F5E0 9430                   JMP $ADDMN
123                     $ZROF2:
124                     ; SET F2 MANTISSA TO 0.
125 F5E2 F0                     LD A, W(X+)     ; X POINTS TO F2-EXP.F2-SIGN.
126 F5E3 00                     CLR A
127 F5E4 F3                     X A, W(X-)      ; AND STORE IT BACK.
128 F5E5 00                     CLR A
129 F5E6 F3                     X A, W(X-)
130 F5E7 00                     CLR A
131 F5E8 F1                     X A, W(X+)
132 F5E9 9427                   JMP $ADDMN
133                     ; F2 EXPONENT IS GREATER THAN F1 EXPONENT.
134                     $F2GTR:
135 F5EB 01                     COMP A
136 F5EC 04                     INC A           ; CHANGE DIFF IN EXP TO POSITIVE.
137 F5ED 80C9CAAB               LD K, H(A)      ; LOAD K WITH LOOP COUNTER.
138 F5F1 8217CAFD               IFGT K, 017
139 F5F5 51                     JP $ZROF1       ; F1 MANT. REDUCED TO 0 IN SHIFTS.
140                     ; LOOP TO SHIFT F1 MANT INTO ALIGNMENT.
141 F5F6 8200CAFC               IFEQ K, 0
142 F5FA 57                     JP $ADDMN       ; K=0 MEANS DONE SHIFTING.
143                     $LOOP1:
144 F5FB E4                     LD A, W(B)
145 F5FC C7                     SHR A
146 F5FD E3                     XS A, W(B-)
147 F5FE 40                     NOP
148 F5FF E4                     LD A, W(B)
149 F600 D7                     RRC A
150 F601 E1                     XS A, W(B+)
151 F602 40                     NOP
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    34
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

152 F603 AACA                   DECSZ K          ;
153 F605 6A                     JP $LOOP1
154 F606 4B                     JP $ADDMN
155                     $ZROF1:                  ; SET F1 MANT TO 0.
156 F607 E0                     LOS A, W(B+)     ; B POINTS TO F1-EXP.F1-SIGN.
157 F608 40                     NOP
158 F609 00                     CLR A
159 F60A E3                     XS A, W(B-)      ; STORE IT BACK.
160 F60B 40                     NOP
161 F60C 00                     CLR A
162 F60D E3                     XS A, W(B-)
163 F60E 40                     NOP
164 F60F 00                     CLR A
165 F610 E1                     XS A, W(B+)
166 F611 40                     NOP
167                     ; DETERMINE IF MANTISSAS ARE TO BE ADDED OR SUBTRACTED.
168                     $ADDMN:                  ; B POINTS TO F1-HI, X TO F2-HI.
169 F612 E0                     LDS A, W(B+)
170 F613 40                     NOP
171 F614 F0                     LD A, W(X+)
172 F615 D4                     LD A, M(X)       ; LOAD F2-SIGN.
173 F616 D8                     XOR A, M(B)      ; XOR WITH F1-SIGN.
174 F617 9C00                   IFEQ A, 0
175 F619 9451                   JMP $TRADD       ; SAME SIGN SO GO TO ADD MANTISSA.
176                     ; GET HERE MEANS TRUE SUBTRACT OF MANTISSA.
177 F61B F2                     LD A, W(X-)
178 F61C F2                     LD A, W(X-)      ; X POINTS TO F2-LO.
179 F61D E2                     LDS A, W(B-)
180 F61E 40                     NOP
181 F61F E2                     LDS A, W(B-)
182 F620 40                     NOP              ; B NOW POINTS TO F1-LO.
183 F621 E0                     LDS A, W(B+)
184 F622 40                     NOP              ; A NOW CONTAINS F1-LO.
185 F623 02                     SET C
186 F624 8FEB                   SUBC A, W(X)     ; SUBTRACT F2-LO.
187 F626 F1                     X A, W(X+)
188 F627 E0                     LDS A, W(B+)     ; A CONTAINS F1-HI.
189 F628 40                     NOP
190 F629 8FEB                   SUBC A, W(X)     ; SUBTRACT F2-HI.
191 F62B F1                     X A, W(X+)
192 F62C 07                     IF C
193 F62D 55                     JP $F1SIN        ; F1 GE F2, SO SIGN IS F1-SIGN.
194                     ; GET HERE MEANS F1 LT F2, SO SIGN IS F2-SIGN.
195 F62E A6FFFCC4A8             LD A, W(SP-4)
196 F633 8FDA                   OR A, M(X)
197 F635 F3                     X A, W(X-)       ; C-EXP.C-SIGN HAS BEEN DETERMINED.
198 F636 F4                     LD A, W(X)
199 F637 D1                     COMP A
200 F638 F3                     X A, W(X-)
201 F639 F4                     LD A, W(X)
202 F63A 01                     COMP A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    35
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

203 F63B B80001                 ADD A, 01
204 F63E F1                     X A, W(X+)
205 F63F 07                     IF C
206 F640 8FA9                   INC W(X)
207 F642 47                     JP $ANORM
208                     ; GET HERE MEANS F1 GE F2.
209                     $F1SIN:
210 F643 A6FFFCC4A8             LD A, W(SP-4)
211 F648 DA                     OR A, M(B)
212 F649 F3                     X A, W(X-)
213                     ; NORMALIZE THE MANTISSA.
214                     $ANORM:
215 F64A ACCECC                 LD B, X         ; B POINTS TO C-HI.
216 F64D E0                     LDS A, W(B+)
217 F64E 40                     NOP
218 F64F C0                     LDS A, M(B+)
219 F650 40                     NOP             ; B NOW POINTS TO C-EXP BYTE.
220 F651 9118                   LD K, 018       ; SET UP LOOP LIMIT OF 24-DEC IN K.
221                     $NLOOP:
222 F653 F4                     LD A, W(X)
223 F654 E7                     SHL A
224 F655 07                     IF C            ; CARRY MEANS NORMALIZED.
225 F656 9448                   JMP $ROUND      ; SO JUMP TO ROUNDING CODE.
226 F658 F3                     X A, W(X-)
227 F659 F4                     LD A, W(X)
228 F65A E7                     SHL A
229 F65B F1                     X A, W(X+)
230 F65C 07                     IF C
231 F65D 8F08                   SET W(X).0
232 F65F ADCC8A                 DECSZ M(B)      ; ADJUST EXPONENT.
233 F662 43                     JP $OV1
234 F663 B4F9B8                 JMPL UNDFL      ; C-EXP ZERO MEANS UNDERFLOW.
235                     $OV1:
236 F666 AACA                   DECSZ K         ; DECREMENT LOOP COUNTER.
237 F668 75                     JP $NLOOP       ; GO BACK TO LOOP.
238 F669 B4F9B2                 JMPL UNDFL      ; UNDERFLOW
239                     ;GET HERE MEANS TRUE ADDITION OF MANTISSA.
240                     $TRADD:
241 F66C E2                     LDS A, W(B-)
242 F66D 40                     NOP
243 F66E E2                     LDS A, W(B-)
244 F66F 40                     NOP             ; B NOW POINTS TO F1-HI.
245 F670 F2                     LD A, W(X-)
246 F671 F2                     LD A, W(X-)
247 F672 F4                     LD A, W(X)      ; LOAD F2-LO INTO A.
248 F673 F8                     ADD A, W(B)     ; ADD F1-LO.
249 F674 F1                     X A, W(X+)      ; STORE IN F2-LO.
250 F675 E0                     LDS A, W(B+)
251 F676 40                     NOP             ; B NOW POINTS TO F1-HI.
252 F677 F4                     LD A, W(X)      ; LOAD F2-HI INTO A.
253 F678 E8                     ADC A, W(B)     ; ADD F1-HI WITH CARRY FROM LO ADD.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    36
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

254 F679 07                         IF C
255 F67A 4A                         JP $ADJEX        ; IF CARRY, NEED TO INCREASE EXP.
256 F67B F1                         X A, W(X+)       ; STORE RESULT IN F2-HI.
257 F67C A6FFFCC4A8                 LD A, W(SP-4)    ; GET C-EXP.00000000.
258 F681 8FDA                       OR A, M(X)       ; INTRODUCE SIGN.
259 F683 F3                         K A, W(X-)       ; STORE IN F2-EXP.F2-SIGN.
260 F684 5B                         JP $ROUND
261                           ; GET HERE MEANS NEED TO INCREASE EXP BY 1.
262                           $ADJEX:
263 F685 D7                         RRC A
264 F686 F3                         X A, W(X-)
265 F687 F4                         LD A, W(X)
266 F688 D7                         RRC A
267 F689 F1                         X A, W(X+)
268 F68A F0                         LD A, W(X+)      ; X NOW POINTS TO F2-EXP.F2-SIGN.
269 F68B A6FFFCC4A8                 LD A, W(SP-4)    ; GET C-EXP.00000000.
270 F690 B80100                     ADD A, 0100      ; INCREASE EXP BY 1.
271 F693 07                         IF C
272 F694 B4F998                     JMPL OVRFL
273 F697 BDFEFF                     IFGT A, 0FEFF    ; IS BIASED EXPONENT 255-DEC ?
274 F69A B4F992                     JMPL OVRFL
275 F69D 8FDA                       OR A, M(X)
276 F69F F3                         X A, W(X-)
277                           ; NEED TO ROUND THE RESULT. X POINTS TO C-HI.
278                           $ROUND:
279 F6A0 B5F9FB                     JSRL SROUND
280                           ; FINAL CHECK OF EXPONENT.
281 F6A3 D0                         LD A, M(X+)      ; X NOW POINTS TO C-EXP.
282 F6A4 D2                         LD A, M(X-)
283 F6A5 9C00                       IFEQ A, 0
284 F6A7 B4F974                     JMPL UNDFL
285 F6AA 90FE                       IFGT A, 0FE
286 F6AC B4F980                     JMPL OVRFL
287 F6AF F2                         LD A, W(X-)
288 F6B0 F2                         LD A, W(X-)      ; X NOW POINTS TO C-LO.
289 F6B1 B5F9C8      +              JSR FPAK         ; PACK C.
290 F6B4 3FC4                       POP SP           ; SET UP SP FOR RETURN.
291 F6B6 3FCC                       POP B
292 F6B8 3FCE                       POP X
293 F6BA 3C                         RET
294                           ;
295                                 .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    37
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FMULT.MAC

 35                                  .FORM 'FMULT.MAC'
 36                                  .INCLD FMULT.MAC
  1                                  .TITLE FMULT
  2                                  .LOCAL
  3                      ;
  4                      ; SUBROUTINE TO MULTIPLY TWO SP FLOATING POINT NUMBERS.
  5                      ;       C = F1*F2
  6                      ;
  7                      ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
  8                      ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
  9                      ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
 10                      ;
 11                      ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
 12                      ; STACK POINTER ON ENTRY, THEN
 13                      ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
 14                      ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
 15                      ;
 16                      ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
 17                      ; REGS. X AND B ARE PRESERVED.
 18                      ;
 19                      FMULT:
 20 F6BB AFCE                    PUSH X          ; SAVE X ON ENTRY.
 21 F6BD AFCC                    PUSH B          ; SAVE B ON ENTRY.
 22                      ; SAVE ADDRESS OF F2-R0 IN TMP1.
 23 F6BF ACC4CE                  LD X, SP
 24 F6C2 86FFF6CEF8              ADD X, 0FFF6    ; SUBTRACT 10.
 25 F6C7 ACCE00                  LD TMP1, X      ; SAVE IN TMP1.
 26                      ; CHECK AND SEE IF F1 IS A NAN.
 27 F6CA B5F973      +           JSR FNACHK
 28 F6CD 07                      IF C
 29 F6CE B4F93E                  JMPL FNAN       ; F1 IS A NAN.
 30                      ; CHECK AND SEE IF F2 IS A NAN.
 31 F6D1 ACCACC                  LD B, K
 32 F6D4 ACC8CE                  LD X, A
 33 F6D7 A20200A8                LD A, W(TMP1+2)
 34 F6DB ACC8CA                  LD K, A
 35 F6DE AECE                    X A, X
 36 F6E0 B5F95D      +           JSR FNACHK
 37 F6E3 07                      IF C
 38 F6E4 B4F928                  JMPL FNAN       ; F2 IS NAN.
 39                      ; CHECK AND SEE IF F2 IS ZERO.
 40 F6E7 B5F967      +           JSR FZCHK
 41 F6EA 07                      IF C
 42 F6EB 94DC                    JMP $CZERO      ; F2 IS ZERO.
 43                      ; CHECK AN SEE IF F1 IS ZERO.
 44 F6ED ACCCCA                  LD K, B         ; RESTORE F1-R1 FROM B.
 45 F6FD B5F95E      +           JSR FZCHK
 46 F6F3 07                      IF C
 47 F6F4 94D3                    JMP $CZERO      ; F1 IS ZERO.
 48                      ; GET HERE MEANS NORMAL MULTIPLICATION.
 49                      ; UNPACK F1 AND F2.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    38
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

  50 F6F6 ACC4CE                LD X, SP        ; X POINTS TO F1-LO.
  51 F6F9 8210C4F8              ADD SP, 010     ; MOVE SP PAST LOCAL STORAGE.
  52 F6FD AFCE                  PUSH X          ; SAVE SP ON STACK FOR QUICK RETURN.
  53 F6FF B5F95F      +         JSR FUNPAK      ; UNPACK F1.
  54 F702 AC00CC                LD B, TMP1      ; B NOW POINTS TO F2-R0.
  55 F705 ACCE00                LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
  56 F708 E0                    LOS A, W(B+)    ; LOAD F2-R0 INTO A.
  57 F709 40                    NOP
  58 F70A AECA                  X A, K
  59 F70C E4                    LD A, W(B)
  60 F70D AECA                  X A, K          ; LOAD F2-R1 INTO K.
  61 F70F B5F94F      +         JSR FUNPAK      ; UNPAK F2.
  62                        ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
  63 F712 F2                    LD A, W(X-)
  64 F713 AC00CC                LD B, TMP1
  65 F716 E2                    LDS A, W(B-)
  66 F717 40                    NOP
  67                        ; COMPUTE C-EXP AND C-SIGN AND STORE IN F2-EXP AND F2-SIGN.
  68 F718 F4                    LD A, W(X)      ; A IS (EEEEEEEE-F2).(SSSSSSSS-F2)
  69 F719 C7                    SHR A           ; SHR SINCE SUM OF EXPS CAN BE 9 BITS.
  70 F71A ACC8CA                LD K, A         ; K IS (DEEEEEEEE-F2).(SSSSSSS-F2)
  71 F71D E4                    LD A, W(B)      ; A IS (EEEEEEEE-F1).(SSSSSSSS-F1)
  72 F71E B9FF00                AND A, OFF00    ; MASK OUT SIGN BITS.
  73 F721 C7                    SHR A           ; A IS (DEEEEEEEE-F1).(0000000)
  74 F722 96CAF8                ADD A, K        ; A IS (EEEEEEEEE-C).(SSSSSSS-F2)
  75 F725 F6                    ST A, W(X)      ; STORE IN F2-SIGN.
  76 F726 E2                    LOS A, W(B-)    ; A IS (EEEEEEEE-F1).(SSSSSSSS-F1)
  77 F727 40                    NOP
  78 F728 99FF                  AND A, OFF      ; MASK OUT EXP BITS.
  79 F72A C7                    SHR A           ; A IS (000000000SSSSSSS-F1)
  80 F72B 8FFB                  XOR A, W(X)     ; A IS (EEEEEEEEESSSSSSS-C)
  81 F72D B8C080                ADD A, 0C080    ; REMOVE EXCESS BIAS OF 127-DEC FROM EXP.
  82 F730 07                    IF C
  83 F731 46                    JP $EXCH2       ; IF CARRY, THEN NO UNDERFLOW NOW
  84                        ; CHECK TO SEE IF EXP IS ZERO. IF NOT, UNDERFLOW FOR SURE.
  85 F732 E7                    SHL A
  86 F733 07                    IF C
  87 F734 B4F8E7                JMPL UNDFL      ; UNDERFLOW, SO JUMP.
  88 F737 C7                    SHR A           ; RESTORE BIT SHIFTED OUT (0).
  89                        ; CHECK FOR EXPONENT OVERFLOW.
  90                        $EXCH2:
  91 F738 E7                    SHL A
  92 F739 07                    IF C            ; IF C IS 1,
  93 F73A B4F8F2                JMPL OVRFL      ; THEN OVERFLOW FOR SURE.
  94 F73D 96C817                IF A.7
  95 F740 96C808                SET A.0         ; RESTORE LAST BIT OF SIGN.
  96 F743 F3                    X A, W(X-)      ; STORE C-EXP. C-SIGN IN F2-EXP.F2-SIGN.
  97                        ;
  98                        ; MULTIPLY THE MANTISSA.
  99                        ; FIRST COMPUTE F1-HI*F2-HI.
 100 F744 F2                    LD A, W(X-)
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    39
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

101 F745 ACCE00                 LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
102 F748 FE                     MULT A, W(B)
103 F749 A6FFFAC4AB             ST A, W(SP-6)   ; STORE LOW WORD OF PRODUCT ON STACK.
104 F74E AECE                   X A, X
105 F750 A6FFFCC4AB             ST A, W(SP-4)   ; STORE HIGH WORD OF PRODUCT ON STACK.
106                         ; NOW COMPUTE F1-HI*F2-LO.
107 F755 AC00CE                 LD X, TMP1
108 F758 F0                     LD A, W(X+)
109 F759 ACCE00                 LD TMP1, X      ; TMP1 NOW POINTS TO F2-HI.
110 F75C FE                     MULT A, W(B)
111 F75D AECE                   X A, X
112 F75F A6FFFAC4F8             ADD A, W(SP-6)  ; ADD LOW WORD OF LAST PROD. TO HIGH WORD.
113 F764 A6FFFAC4AB             ST A, W(SP-6)
114 F769 07                     IF C
115 F76A A6FFFCC4A9             INC W(SP-4)     ; IF CARRY, INCREASE HIGH WORD BY 1.
116                         ; FINALLY COMPUTE F1-LO*F2-HI.
117 F76F E2                     LDS A, W(B-)    ; ADJUST B TO POINT TO F1-LO.
118 F770 40                     NOP
119 F771 AC00CE                 LD X, TMP1
120 F774 F4                     LD A, W(X)
121 F775 FE                     MULT A, W(B)
122 F776 AECE                   X A, X
123 F778 A6FFFAC4F8             ADD A, W(SP-6)  ; ADD LOW WORD ACCUMULATED SO FAR.
124 F77D A6FFFAC4AB             ST A, W(SP-6)
125 F782 A6FFFCC4AB             LD A, W(SP-4)   ; A CONTAINS HIGH WORD OF PRODUCT.
126 F787 07                     IF C            ; IF CARRY ON LAST LOW WORD ADD,
127 F788 04                     INC A           ; THEN INCREASE HIGH WORD.
128                         ;
129                         ; MANTISSA MULTIPLICATION DONE. NOW CHECK FOR NORMALIZATION.
130 F789 AC00CE                 LD X, TMP1
131 F78C BD7FFF                 IFGT A, 07FFF   ; IS MSB OF PRODUCT 1 ?
132 F78F 4D                     JP $EXINC       ; YES, INCREASE MANTISSA.
133                                             ; NEED TO SHIFT MANTISSA LEFT BY 1 BIT.
134 F790 E7                     SHL A
135 F791 F3                     X A, W(X-)
136 F792 A6FFFAC4AB             LD A, W(SP-6)
137 F797 E7                     SHL A
138 F798 F1                     X A, W(X+)
139 F799 07                     IF C            ; DID SHIFT OF LOW WORD PUSH OUT A 1 ?
140 F79A 8F08                   SET W(X).0      ; YES SO SET LSB OF HIGH WORD.
141 F79C 51                     JP $ROUND       ; GO TO ROUNDING CODE.
142                         $EXINC:
143                         ; NEED TO INCREASE EXPONENT BY 1. REMEMBER X POINTS TO F2-HI.
144 F79D F3                     X A, W(X-)      ; A CONTAINS HIGH WORD, X POINTS TO F2-LO.
145 F79E A6FFFAC4A8             LD A, W(SP-6)   ; GET LOW WORD.
146 F7A3 F1                     X A, W(X+)      ; STORE LOW WORD.
147 F7A4 F0                     LD A, W(X+)
148 F7A5 F4                     LD A, W(X)      ; GET C-EXP.C-SIGN
149 F7A6 B80100                 ADD A, 0100     ; INCREASE C-EXP.
150 F7A9 07                     IF C
151 F7AA B4F882                 JMPL OVRFL      ; EXPONENT OVERFLOW.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    40
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

152 F7AD F3                     X A, W(X-)      ; NO OVERFLOW, SO SAVE C-EXP.C-SIGN.
153                     ; ROUNDING CODE.
154                     $ROUND:
155 F7AE B5F8ED                 JSRL SROUND
156                     ; FINAL CHECK OF EXPONENT.
157 F7B1 D0                     LD A, M(X+)     ; X NOW POINTS TO C-EXP.
158 F7B2 D2                     LD A, M(X-)
159 F7B3 9C00                   IFEQ A, 0
160 F7B5 B4F866                 JMPL UNDFL
161 F7B8 9DFE                   IFGT A, OFE
162 F7BA B4F872                 JMPL OVRFL
163 F7BD F2                     LD A, W(X-)
164 F7BE F2                     LD A, W(X-)     ; X NOW POINTS TO C-LO.
165 F7BF B5F8BA      +          JSR FPAK        ; PACK C.
166 F7C2 3FC4                   POP SP          ; SET UP SP FOR RETURN.
167 F7C4 3FCC                   POP B
168 F7C6 3FCE                   POP X
169 F7C8 3C                     RET
170                     ; EXCEPTION HANDLING.
171                     ; C IS ZERO B'COS ONE OF F1 OR F2 IS ZERO.
172                     $CZERO:
173 F7C9 00                     CLR A
174 F7CA ACC8CA                 LD K, A
175 F7CD 3FCC                   POP B
176 F7CF 3FCE                   POP X
177 F7D1 3C                     RET
178                     ;
179                             .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    41
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FDIV.MAC

                                    .FORM 'FDIV.MAC'
                                    .INCLD FDIV.MAC
                                    .TITLE FDIV
                                    .LOCAL
                        ;
                        ; SUBROUTINE TO DIVIDE TWO SP FLOATING POINT NUMBERS.
                        ;    C = F1/F2
                        ;
                        ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                        ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                        ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                        ;
                        ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                        ; STACK POINTER ON ENTRY, THEN
                        ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                        ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                        ;
                        ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                        ;
                       FDIV:
F7D2 AFCE                       PUSH X
F7D4 AFCC                       PUSH B
                        ; SAVE ADDRESS OF F2-R0 IN TMP1.
F7D6 ACC4CE                     LD X, SP
F7D9 86FFF6CEF8                 ADD X, 0FFF6    ; SUBTRACT 10.
F7DE ACCE00                     LD TMP1, X      ; AND SAVE IN TMP1.
                        ; CHECK AND SEE IF F1 IS A NAN.
F7E1 85F85C             +       JSR FNACHK
F7E4 07                         IF C
F7E5 B4F827                     JMPL FNAN       ; F1 IS A NAN.
                        ; CHECK AND SEE IF F2 IS A NAN.
F7E8 ACCACC                     LD B, K
F7EB ACC8CE                     LD X, A
F7EE A20200A8                   LD A, W(TMP1+2)
F7F2 ACC8CA                     LD K, A
F7F5 AECE                       X A. X
F7F7 B5F846             +       JSR FNACHK
F7FA 07                         IF C
F7FB B4F811                     JMPL FNAN       ; F2 IS NAN.
                        ; CHECK AND SEE IF F2 IS ZERO.
F7FE B5F850             +       JSR FZCHK
F801 07                         IF C
F802 B4F7FB                     JMPL DIVBY0     ; F2 IS ZERO.
                        ; CHECK AND SEE IF F1 IS ZERO.
F805 ACCCCA                     LD K, B         ; RESTORE F1-R1 FROM B.
F808 B5F846             +       JSR FZCHK
F80B 07                         IF C
F80C 94F1                       JMP $CZERO      ; F1 IS ZERO.
                        ; GET HERE MEANS NORMAL DIVISION.
                        ; UNPACK F1 AND F2.
F80E ACC4CE                     LD X, SP        ; X POINTS TO F1-L0.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    42
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

  50 F811 8210C4F8              ADD SP, 010      ; MOVE SP PAST LOCAL STORAGE.
  51 F815 AFCE                  PUSH X           ; SAVE SP ON STACK FOR QUICK RETURN.
  52 F817 B5F847      +         JSR FUNPAK       ; UNPACK F1.
  53 F81A AC00CC                LD B, TMP1       ; B NOW POINTS TO F2-R0
  54 F81D ACCE00                LD TMP1, X       ; TMP1 NOW POINTS TO F2-L0.
  55 F820 E0                    LDS A, W(B+)     ; LOAD F2-R0 INTO A.
  56 F821 40                    NOP
  57 F822 AECA                  X A, K
  58 F824 E4                    LD A, W(B)
  59 F825 AECA                  X A, K           ; LOAD F2-R1 INTO K.
  60 F827 B5F837      +         JSR FUNPAK       ; UNPAK F2.
  61                    ;
  62                    ; ENSURE THAT F1-HI IS LESS THAN F2-HI.
  63                    ;
  64 F82A F2                    LD A, W(X-)      ; X POINTS TO F2-EXP.F2-SIGN.
  65 F82B F2                    LD A, W(X-)      ; X POINTS TO F2-HI.
  66 F82C AC00CC                LD B, TMP1       ; B POINTS TO F2-L0.
  67 F82F E2                    LDS A, W(B-)     ; B POINTS TO F1-EXP.F1-SIGN.
  68 F830 40                    NOP
  69 F831 E2                    LDS A, W(B-)     ; LOAD F1-EXP.F1-SIGN.
  70 F832 40                    NOP              ; B POINTS TO F1-HI.
  71 F833 ACC8CA                LD K, A          ; SAVE F1-EXP.F1-SIGN IN K.
  72 F836 F4                    LD A, W(X)       ; LOAD F2-HI.
  73 F837 FD                    IFGT A, W(B)     ; IS F2-HI > F1-HI ?
  74 F838 51                    JP $FEXSN        ; YES, SO ALL IS WELL.
  75                                             ; GET HERE MEANS NEED TO SHR F1,
  76                                             ; AND INCREASE ITS EXPONENT.
  77 F839 E0                    LOS A, W(B+)     ; GET F1-HI.
  78 F83A 40                    NOP              ; B POINTS TO F1-EXP.F1-SIGN.
  79 F83B AECA                  X A, K           ; SWAP F1-EXP.F1-SIGN AND F1-HI.
  80 F83D B80100                ADD A, 0100      ; INCREASE F1-EXP BY 1.
  81 F840 E3                    XS A, W(B-)      ; STORE BACK IN F1-EXP.F1-SIGN.
  82 F841 40                    NOP              ; B POINTS TO F1-HI.
  83 F842 E4                    LD A, W(B)       ; LOAD F1-HI.
  84 F843 C7                    SHR A
  85 F844 E3                    XS A, W(B-)      ; STORE BACK IN F1-HI.
  86 F845 40                    NOP              ; B POINTS TO F1-L0.
  87 F846 E4                    LD A, W(B)       ; LOAD F1-L0.
  88 F847 D7                    RRC A
  89 F848 E1                    XS A, W(B+)      ; PUT IT BACK IN F1-L0.
  90 F849 40                    NOP              ; B POINTS TO F1-HI.
  91                    ;
  92                    $FEXSN:
  93                    ; DETERMINE C-EXP AND C-SIGN.
  94 F84A F0                    LD A, W(X+)      ; X POINTS TO F2-EXP.F2-SIGN.
  95 F84B E0                    LDS A, W(B+)     ; B POINTS TO F1-EXP.F1-SIGN.
  96 F84C 40                    NOP
  97 F84D F4                    LD A, W(X)       ; LOAD F2-EXP.F2-SIGN.
  98 F84E B9FF00                AND A, 0FF00     ; MASK OUT THE SIGN.
  99 F851 C7                    SHR A            ; ALLOW 9 BITS FOR EXP CALCULATIONS.
 100 F852 ACC8CA                LD K, A          ; SAVE IT IN K.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    43
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

101 F855 E4                     LD A, W(B)      ; LOAD F1-EXP.F1-SIGN.
102 F856 B9FF00                 AND A, 0FF00    ; MASK OUT SIGN.
103 F859 C7                     SHR A
104 F85A 02                     SET C
105 F85B 96CAEB                 SUBC A, K       ; SUBTRACT THE EXPONENTS.
106                                             ; NOTE THAT NOW THE MS 9 BITS
107                                             ; OF A CONTAIN A 2'S COMP. INTEGER.
108 F85E B7000000               LD TMP1, 0
109 F862 E7                     SHL A
110 F863 07                     IF C
111 F864 B700FF00               LD TMP1, 0FF
112 F868 D7                     RRC A           ; SAVE SIGN OF NUMBER IN TMP1.
113 F869 B83F00                 ADD A, 03F00    ; RESTORE IEEE BIAS.
114 F86C E7                     SHL A           ; MAKE EXPONENT 8 BITS.
115 F86D 06                     IFN C           ; NO CARRY ?
116 F86E 49                     JP $FSIGN       ; THEN ALL IS WELL.
117 F86F 960010                 IF TMP1.0       ; WAS EXP NEGATIVE BEFORE ?
118 F872 B4F7A9                 JMPL UNDFL      ; YES, SO UNDERFLOW.
119 F875 B4F7B7                 JMPL OVRFL      ; OTHERWISE OVERFLOW.
120                     ;
121                     $FSIGN:
122                     ; C-EXP HAS BEEN COMPUTED. NOW FIND C-SIGN.
123                     ; BUT FIRST TAKE CARE OF SPECIAL OVER/UNDERFLOW CASES.
124 F878 BCFF00                 IFEQ A, 0FF00
125 F87B B4F7B1                 JMPL OVRFL
126 F87E 9C00                   IFEQ A, 0
127 F880 B4F79B                 JMPL UNDFL
128 F883 AB00                   ST A, TMP1      ; SAVE C-EXP.00000000 IN TMP1.
129 F885 F4                     LD A, W(X)      ; LOAD F2-EXP.F2-SIGN.
130 F886 99FF                   AND A, 0FF      ; MASK OUT F2-EXP.
131 F888 FB                     XOR A, W(B)     ; A NOW HAS F1-EXP.C-SIGN.
132 F889 99FF                   AND A, 0FF      ; MASK OUT F1-EXP.
133 F88B 9600FA                 OR A, TMP1      ; BRING IN C-EXP.
134 F88E F3                     X A, W(X-)      ; STORE IN F2-EXP.F2-SIGN.
135                                             ; X POINTS TO F2-HI.
136 F88F F2                     LD A, W(X-)     ; X POINTS TO F2-LO.
137 F890 E2                     LDS A, W(B-)    ; B POINTS TO F1-HI.
138 F891 40                     NOP
139                     ;
140                     ; NOW DO THE MANTISSA DIVISION.
141                     ;
142 F892 F0                     LD A, W(X+)     ; LOAD F2-LO. X POINTS TO F2-HI.
143 F893 ACCE00                 LD TMP1, X      ; SAVE ADDRESS OF F2-HI IN TMP1.
144 F896 FE                     MULT A, W(B)    ; COMPUTE F2-LO*F1-HI.
145                                             ; X CONTAINS MS WORD AND A IS LS WORD.
146                     ;
147 F897 AECC                   X A, B          ; A POINTS TO F1-HI, B CONTAINS LS WORD.
148 F899 AE00                   X A, TMP1       ; A POINTS TO F2-HI, TMP1 POINTS TO F1-HI.
149 F89B AECC                   X A, B          ; A CONTAINS LS WORD, B POINTS TO F2-HI.
150                     ;
151 F890 EF                     .BYTE 0EF       ; DIVD A, W(B) - KLUDGED !!
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    44
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

152                         ;
153 F89E ACC8CA                     LD K, A          ; SAVE QUOTIENT IN K.
154 F8A1 A8CC                       LD A, B          ; A POINTS TO F2-HI.
155 F8A3 AE00                       X A, TMP1        ; A POINTS TO F1-HI, TMP1 POINTS TO F2-HI.
156 F8A5 AECC                       X A, B           ; B POINTS TO F1-HI.
157 F8A7 E2                         LDS A, W(B-)     ; B POINTS TO F1-LO.
158 F8A8 40                         NOP
159 F8A9 E0                         LOS A, W(B+)     ; LOAD F1-LO.
160 F8AA 40                         NOP              ; B POINTS TO F1-HI.
161 F8AB 02                         SET C
162 F8AC 96CAEB                     SUBC A, K        ; SUBTRACT QUOTIENT SAVED IN K.
163 F8AF ACC8CE                     LD X, A          ; AND SAVE IN X.
164 F8B2 E4                         LD A, W(B)       ; LOAD F1-HI.
165 F8B3 06                         IFN C            ; IF C WAS NOT SET IN THE LAST SUBTRACT,
166 F8B4 05                         DEC A            ; ADJUST THE BORROW.
167 F8B5 AECE                       X A, X
168 F8B7 AC00CC                     LD B, TMP1       ; B POINTS TO F2-HI.
169                         ;
170 F8BA EF                         .BYTE 0EF        ; DIVD A, W(B) - KLUDGED AGAIN !
171                                                  ; QUOTIENT IN A, REM IN X.
172                         ;
173 F8BB AB00                       ST A, TMP1       ; SAVE QUOTIENT IN TMP1.
174 F8BD 00                         CLR A            ; ZERO A.
175                         ;
176 F8BE EF                         .BYTE 0EF        ; DIVD A, W(B) - KLUDGED YET AGAIN !
177                         ;
178 F8BF AE00                       X A, TMP1        ; SWAP OLD AND NEW QUOTIENTS.
179                         ;
180                         ; CHECK FOR NORMALIZATION. CAN BE OFF BY AT MOST 1 BIT.
181 F8C1 E7                         SHL A
182 F8C2 07                         IF C
183 F8C3 56                         JP $NMED         ; IT IS NORMALIZED.
184                                                  ; GET HERE MEANS NEED TO SHIFT LEFT ONCE.
185 F8C4 AE00                       X A, TMP1        ; SWAP HI AND LO WORDS.
186 F8C6 E7                         SHL A
187 F8C7 AE00                       X A, TMP1        ; HI WORD IS IN A, LO WORD IN TMP1.
188 F8C9 07                         IF C             ; WAS 1 SHIFTED OUT OF LO WORD?
189 F8CA 96C808                     SET A.0          ; YES, THEN SET LSB OF HI WORD.
190 F8CD ABCA                       ST A, K          ; SAVE HI WORD IN K.
191 F8CF E1                         XS A, W(B+)
192 F8D0 40                         NOP              ; B POINTS TO F2-EXP.F2-SIGN.
193 F8D1 E4                         LD A, W(B)       ; LOAD F2-EXP.F2-SIGN.
194 F8D2 B8FF00                     ADD A, 0FF00     ; SUBTRACT 1 FROM EXPONENT.
195 F8D5 E3                         XS A, W(B-)      ; STORE BACK IN F2-EXP.F2-SIGN.
196 F8D6 40                         NOP              ; B POINTS TO F2-HI.
197 F8D7 A8CA                       LD A, K          ; HI WORD TO A.
198 F8D9 E7                         SHL A
199                         $NMED:
200 F8DA D7                         RRC A            ; RESTORE BIT OUT.
201 F8DB E3                         XS A, W(B-)      ; SAVE HI-WORD IN F2-HI.
202 F8DC 40                         NOP              ; B POINTS TO F2-LO.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    45
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

203 F8DD A800                       LD A, TMP1
204 F8DF E1                         XS A, W(B+)      ; SAVE C-LO.
205 F8E0 40                         NOP              ; B POINTS TO F2-HI.
206 F8E1 ACCCCE                     LD X, B          ; MOVE ADDRESS OF F2-HI TO X.
207                         ;
208                         ; ROUNDING CODE.
209 F8E4 B5F7B7                     JSRL SROUND
210                         ; FINAL CHECK OF EXPONENT.
211 F8E7 D0                         LD A, M(X+)      ; X NOW POINTS TO C-EXP.
212 F8E8 D2                         LD A, M(X-)
213 F8E9 9C00                       IFEQ A, 0
214 F8EB B4F730                     JMPL UNDFL
215 F8EE 9DFE                       IFGT A, 0FE
216 F8F0 B4F73C                     JMPL OVRFL
217 F8F3 F2                         LD A, W(X-)
218 F8F4 F2                         LD A, W(X-)      ; X NOW POINTS TO C-LO.
219 F8F5 B5F784      +              JSR FPAK         ; PACK C.
220 F8F8 3FC4                       POP SP           ; SET UP SP FOR RETURN.
221 F8FA 3FCC                       POP B
222 F8FC 3FCE                       POP X
223 F8FE 3C                         RET
224                         ; C IS ZERO B'COS F1 IS ZERO.
225                         $CZERO:
226 F8FF 00                         CLR A
227 F900 ACC8CA                     LD K, A
228 F903 3FCC                       POP B
229 F905 3FCE                       POP X
230 F907 3C                         RET
231                         ;
232                                 .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    46
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FSINX.MAC

  39                                    .FORM 'FSINX.MAC'
  40                                    .INCLD FSINX.MAC
   1                    ;
   2                                    .TITLE SINX
   3                                    .LOCAL
   4                    ; A VERY DIRTY APPROXIMATION TO SIN(X).
   5                    ; X SHOULD BE IN RADIANS.
   6                    ;
   7                    ; ON INPUT X SHOULD BE IN IEEE FLP FORMAT IN REGS. K AND A.
   8                    ; ON RETURN SIN(X) IS IN IEEE FLP FORMAT IN REGS. K AND A.
   9                    ;
  10                    SINX:
  11 F908 AFCE                  PUSH X          ; SAVE X.
  12 F90A AFC8                  PUSH A
  13 F90C AFCA                  PUSH K          ; X TO THE STACK.
  14 F90E 3653          -       JSRL FMULT      ; COMPUTE X^2.
  15 F910 AFC8                  PUSH A
  16 F912 AFCA                  PUSH K          ; X^2 TO THE STACK.
  17 F914 B6F994A8              LD A, W($A5LO)
  18 F918 A4F996CAAB            LD K, W($A5HI)  ; LOAD A5.
  19 F91D 3662          -       JSRL FMULT      ; COMPUTE A5*X^2.
  20 F91F AFC8                  PUSH A
  21 F921 AFCA                  PUSH K
  22 F923 B6F998A8              LD A, W($A4LO)
  23 F927 A4F99ACAAB            LD K, W($A4HI)  ; LOAD A4.
  24 F92C B5FBE6                JSRL FSUB       ; COMPUTE A4-A5*X^2.
  25 F92F 3FCE                  POP X
  26 F931 3FCE                  POP X
  27 F933 3678          -       JSRL FMULT      ; COMPUTE
  28                                            ; X^2(A4 - A5*X^2).
  29 F935 AFC8                  PUSH A
  30 F937 AFCA                  PUSH K
  31 F939 B6F99CA8              LD A, W($A3LO)
  32 F93D A4F99ECAAB            LD K, W($A3HI)  ; LOAD A3.
  33 F942 B5FBD0                JSRL FSUB       ; COMPUTE
  34                                            ; A3 - X^2(A4 - A5*X^2).
  35 F945 3FCE                  POP X
  36 F947 3FCE                  POP X
  37 F949 368E          -       JSRL FMULT      ; COMPUTE
  38                                            ; X^2(A3 - X^2(A4 - A5*X^2)).
  39 F94B AFC8                  PUSH A
  40 F94D AFCA                  PUSH K
  41 F94F B6F9A0A8              LD A, W($A2LO)
  42 F953 A4F9A2CAAB            LD K, W($A2HI)  ; LOAD A2.
  43 F958 B5FBBA                JSRL FSUB       ; COMPUTE
  44                                            ; A2 - X^2(A3 - X^2(A4 - A5*X^2)).
  45 F95B 3FCE                  POP X
  46 F95D 3FCE                  POP X
  47 F95F 36A4          -       JSRL FMULT      ; COMPUTE
  48                                            ; X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
  49 F961 AFC8                  PUSH A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    47
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
FSINX.MAC

  50 F963 AFCA                  PUSH K
  51 F965 B6F9A4AB              LD A, W($A1LO)
  52 F969 A4F9A6CAAB            LD K, W($A1HI)   ; LOAD A1.
  53 F96E B5FBA4                JSRL FSUB        ; COMPUTE
  54                                             ; A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
  55 F971 3FCE                  POP X
  56 F973 3FCE                  POP X
  57 F975 36BA          -       JSRL FMULT       ; COMPUTE
  58                                             ; X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2)))).
  59 F977 AFC8                  PUSH A
  60 F979 AFCA                  PUSH K
  61 F97B B13F80                LD K, 03F80
  62 F97E 00                    CLR A            ; LOAD 1.0 INTO K-A.
  63 F97F B5FB93                JSRL FSUB        ; COMPUTE
  64                                             ; 1 - ALL THE JUNK ABOVE.
  65 F982 3FCE                  POP X
  66 F984 3FCE                  POP X
  67 F986 3FCE                  POP X
  68 F988 3FCE                  POP X            ; NOW X IS AT THE TOP OF STACK.
  69 F98A 36CF          -       JSRL FMULT       ; COMPUTE
  70                                     ; X(1 - X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))))).
  71 F98C 3FCE                  POP X
  72 F98E 3FCE                  POP X
  73 F990 3FCE                  POP X
  74 F992 3C                    RET
  75                    ;
  76 F993 40                            .EVEN
  77                    ;
  78 F994 2B32          $A5LO: .WORD 0322B
  79 F996 D732          $A5HI: .WORD 032D7
  80 F998 1DEF          $A4LO: .WORD 0EF1D
  81 F99A 3836          $A4HI: .WORD 03638
  82 F99C 010D          $A3LO: .WORD 00D01
  83 F99E 5039          $A3HI: .WORD 03950
  84 F9A0 8988          $A2LO: .WORD 08889
  85 F9A2 083C          $A2HI: .WORD 03C08
  86 F9A4 ADAA          $A1LO: .WORD 0AAAD
  87 F9A6 2A3E          $A1HI: .WORD 03E2A
  88                    ;
  89                    ; A DIRTY APPROXIMATION TO COS(X) USING SIN(X).
  90                    ;
  91                    COSX:
  92 F9A8 AFCE                  PUSH X
  93 F9AA ACC8CE                LD X, A
  94 F9AD B6F9C8A8              LD A, W($PI2LO)
  95 F9B1 AFC8                  PUSH A
  96 F9B3 B6F9CAA8              LD A, W($PI2HI)
  97 F9B7 AFC8                  PUSH A
  98 F9B9 A8CE                  LD A, X
  99 F9BB B5FB77                JSRL FADD        ; COMPUTE X + PI/2.
 100 F9BE 3FCE                  POP X
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    48
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
FSINX.MAC

101 F9C0 3FCE                       POP X
102 F9C2 34BA             -         JSRL SINX        ; COMPUTE SIN(X+PI/2).
103 F9C4 3FCE                       POP X
104 F9C6 3C                         RET
105                       ;
106 F9C7 40                                 .EVEN
107 F9C8 DB0F             $PI2L0: .WORD 00FDB
108 F9CA C93F             $PI2HI: .WORD 03FC9
109                       ;
110                       ; A DIRTY APPROXIMATION TO TAN(X) USING SINX AND COSX.
111                       ;
112                       TANX:
113 F9CC AFCE                       PUSH X
114 F9CE AFCC                       PUSH B
115 F9D0 AFC8                       PUSH A
116 F9D2 AFCA                       PUSH K
117 F9D4 342C                       JSR COSX         ; COMPUTE COS(X)
118 F9D6 ACC8CE                     LD X, A
119 F9D9 ACCACC                     LD B, K
120 F9DC 3FCA                       POP K
121 F9DE 3FC8                       POP A
122 F9E0 AFCE                       PUSH X
123 F9E2 AFCC                       PUSH B
124 F9E4 34DC                       JSR SINX         ; COMPUTE SIN(X).
125 F9E6 3614                       JSR FDIV         ; COMPUTE TAN(X) = SIN(X)/COS(X).
126 F9E8 3FCC                       POP B
127 F9EA 3FCC                       POP B
128 F9EC 3FCC                       POP B
129 F9EE 3FCE                       POP X
130 F9F0 3C                         RET
131                       ;
132                                 .END
 41                       ;
 42                       ;
 43 FFFE 00F0                              .END LISTER
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    49
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
SYMBOL TABLE

A       00C8 W      ATOF    F13A *      B       00CC W      BFMUL   F0C7
COSX    F9A8        DIVBY0  F000        FADD    F535        FDIV    F7D2
FMULT   F6BB        FNACHK  F040        FNAN    F00F        FPAK    F07C
FPERWD  0002 W      FPTRAP  F09D        FSUB    F515        FTOA    F311 *
FUNPAK  F061        FZCHK   F051        ISIOK   F105        K       00CA W
LISTER  F000        MUL10   F118        OVRFL   F02F        PC      00C6 W
SINX    F908        SP      00C4 W      SROUND  F09E        TANX    F9CC *
TMP1    0000 W      UNDFL   F01E        X       00CE W      $A10EX  F1F4
$A1HI   F9A6        $A1LO   F9A4        $A2HI   F9A2        $A2LO   F9A0
$A3HI   F99E        $A3LO   F99C        $A4HI   F99A        $A4LO   F998
$A5HI   F996        $A5LO   F994        $ACCF   F1A9        $ACCM   F177
$ADDEX  F1FF        $ADDMN  F612        $ADEM   F3EC        $ADJEX  F685
$ANORM  F64A        $ANOTO  F05C        $CHKOT  F113        $CHNGS  F361
$CSIGN  F349        $CZERO  F7C9        $CZERO  F8FF        $DIV10  F272
$DIV10  F370        $DOLUP  F481        $DTHI   F270        $DTHI   F392
$DTLO   F26E        $DTLO   F390        $ESAVE  F20F        $ESIGN  F1C7
$EXACC  F1C9        $EXCH2  F738        $EXCHR  F1BB        $EXCLP  F1D4
$EXCOL  F1BA        $EXCPT  F2CF        $EXIN2  F0B7        $EXINC  F79D
$EXIT   F0C6        $F1CHK  F56E        $F1SIN  F643        $F2GTR  F5EB
$TEXSN  F84A        $FRCOL  F18B        $FSIGN  F878        $GOBAK  F41F
$GOON   F42B        $HIUP   F0AF        $INCOL  F15C        $INCRV  F40C
$INDUN  F445        $INTFY  F43C        $ISNAN  F04C        $ISNED  F298
$ISNED  F3BE        $ISNXT  F17D        $JAMDN  F29E        $JAMDN  F3C4
$JAMIT  F280        $JAMIT  F3A6        $JAMLP  F287        $JAMLP  F3AD
$LOOP1  F147        $LOOP1  F5FB        $LOOP2  F5D7        $ML4    F3EA
$MLOG2  F33F        $MSIGN  F154        $MTHI   F26C        $MTHI   F396
$MTLO   F26A        $MTLO   F394        $MUL10  F398        $NAGAS  F2C3
$NAN    F4CF        $NANLP  F4DC        $NEG10  F20B        $NLOOP  F653
$NMED   F8DA        $NORM1  F230        $NORM2  F235        $NRDUN  F247
$NRLUP  F237        $NTZER  F585        $OV1    F666        $OVR1   F29B
$OVR1   F3C1        $PI2HI  F9CA        $PI2LO  F9C8        $REMV9  F354
$RNDUP  F0A7        $ROUND  F6A0        $ROUND  F7AE        $TRADD  F66C
$VDOWN  F418        $VUP    F416        $ZERLP  F505        $ZERO   F4E9
$ZROF1  F607        $ZROF2  F5E2
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    50
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
MACRO TABLE

   NO WARNING LINES

   NO ERROR LINES

  2547 ROM BYTES USED

SOURCE CHECKSUM = A31F
OBJECT CHECKSUM = 2AC3

INPUT  FILE C:LISTER.MAC
LISTING FILE C:LISTER.PRN
OBJECT  FILE C:LISTER.LM
```

A Radix 2 FFT Program for the HPC

National Semiconductor
Application Note 487
Ashok Krishnamurthy

INTRODUCTION

This report describes the implementation of a radix-2, Decimation-in-time FFT algorithm on the HPC. The program, as presently set up can do FFTs of length 2, 4, 8, 16, 32, 64, 128 and 256. The program can be easily modified to work with higher FFT lengths by increasing the Twiddle Factor table.

FFT FUNDAMENTALS

If x(n), n = 0, 1, . . . , N−1 are N samples of a time domain signal, its Discrete Fourier Transform (DFT) is defined as

$$X(k) = \sum_{n=0}^{n=N-1} x(n)\, W^{nk}, \quad k = 0, 1, \ldots, N-1$$

where $W = e^{-j2\pi/N}$

The straight evaluation of the above equation requires on the order of N^2 complex multiplies. The FFT is nothing but a fast algorithm to compute the DFT that uses only on the order of N log(N) complex multiplies. Many different FFT algorithms exist (please see references 1, 2 and 3). The algorithm implemented for the HPC is the most common type of FFT — a radix-2, Decimation-in-time algorithm. This class of algorithms requires that the number of input samples, N, be a power of 2. This is usually not a problem, since the input data can be zero padded to achieve this. The development of this algorithm is described in references 1 and 2; the discussion here is brief and based on reference 1.

Separating the DFT summation above into the even-numbered points and odd-numbered points of x(n), we can rewrite the above sum as:

$$X(k) = \sum_{n\ \text{even}} x(n)\, W^{nk} + \sum_{n\ \text{odd}} x(n)\, W^{nk}$$

Using n = 2r for n even and n = 2r + 1 for n odd, we can further rewrite the above as:

$$X(k) = \sum_{r=0}^{N/2-1} x(2r)\, W^{2rk} + W^k \sum_{r=0}^{N/2-1} x(2r+1)\, W^{2rk}$$

If G(k) is the N/2 point DFT of x(2r) and H(k) is the N/2 point DFT of x(2r+1), the above equation can be written as:

$$X(k) = G(k) + W^k H(k)$$

This equation shows that a N point DFT can be written as the sum of two N/2 point DFTs. The N/2 point DFTs can be computed as the sum two N/4 point DFTs and so on until we are left with two point DFTs. The two point DFTs can be trivially evaluated by direct computation.

Figure 1, taken from reference 1, shows the decomposition for the case N = 8. With reference to this figure, we can note the following points.

1. If N is the number of points in the original sequence, where $N = 2^L$, then there are L stages in the DFT decomposition.
2. The basic computation unit is the so-called Butterfly, shown in *Figure 2*. Each stage involves the computation of N/2 butterflies.
3. The results from the computation in one stage are fed to the next stage after multiplication by some power of W. These powers of W are the so-called Twiddle Factors. Note that each power of W is really a complex number that can be represented by its real and imaginary parts. The real part of W^k is $\cos(2\pi k/N)$ and the imaginary part is $-\sin(2\pi k/N)$.
4. The number of distinct Twiddle Factors used in the first stage is 1, in the second stage is 2 etc., until the L^{th} stage that involves $2^{L-1} = N/2$ twiddle factors. Each twiddle factor in the first stage is involved in N/2 Butterflies, in the second stage with N/4 butterflies etc., until in the L^{th} stage each twiddle factor is involved with $N/(2^L) = 1$ butterfly.
5. The input data sequence needs to be suitably scrambled if the output sequence is to be in the proper order. This scrambling is easily accomplished by using the so-called Bit-Reverse counter as outlined in reference 2.
6. The outputs from each stage can be stored back again in the same storage area as the input sequence. This gives the algorithm the in-place property. Thus the final DFT results overwrite the initial data.

THE INVERSE FFT

If X(k) k = 0, 1, . . . , N−1 is the DFT of a sequence, then its inverse DFT, x(n), is defined as follows:

$$x(n) = \left(\frac{1}{N}\right) \sum_{k=0}^{k=N-1} X(k)\, W^{-nk} \quad n = 0, 1, .., N-1.$$

Thus the Inverse FFT is the same as the forward FFT except for the following: 1. Negative powers of W are used instead of positive powers; and 2. The final sequence is scaled by 1/N. The basic FFT program can therefore be used to compute the inverse FFT with these two changes. This is the approach used in the HPC implementation.

TWIDDLE FACTOR TABLE

The brief description of the FFT in the previous section shows that the algorithm needs to use the Twiddle Factors W^k in the computation. The twiddle factors can either be computed as required, they can be computed using a recursive relation, or they can be obtained by looking up in a table (Ref. 2). The approach used in the HPC implementation is to construct a table containing the needed twiddle factors. This table is stored in ROM and values needed are looked up from this table. The length of the table needed is determined by the maximum FFT length that you want to use. The HPC FFT implementation is presently limited to a maximum length of 256. This requires that the twiddle factors W^0, W^1, . . . W^{255} be available, where

$W = e^{-j2\pi/256}$. Since $e^{jx} = \cos(x) + j\sin(x)$, the values stored in this table are cos(0), sin(0), cos(2π/256), sin(2π/256) etc., up to cos($2\pi \times 255/256$), sin ($2\pi \times 255/256$). The table used in the implementation is organized as follows:

.WORD cos(0) × 2^{14}
.WORD sin(0) × 2^{14}
.WORD cos(2π/256) × 2^{14}
.WORD sin(2π/256) × 2^{14}
.
.
.
.WORD cos(2π255/256) × 2^{14}
.WORD sin(2π255/256) × 2^{14}

This table is available in the file TWDTBL.MAC and occupies 1024 bytes of storage.

DATA STORAGE

The data to be transformed, x(0), . . . , x(N−1) are also regarded as complex numbers with a real and an imaginary part. Let xr(i) be the real part of x(i) and xi(i) the imaginary part of x(i). Then the data needs to be stored as follows:

.WORD xr(0)
.WORD xi(0)
.WORD xr(1)
.WORD xi(1)
.
.
.
.WORD xr(N−1)
.WORD xi(N−1)

The length of this storage area obviously depends on the number of data points to be transformed. Note that the FFT program itself does not use any base page user RAM. Also, only 8 words of stack are needed. Thus the base page user RAM can be used to store the data to be transformed. Since 192 bytes are available in this area, transforms of up to 32 point in length can be in the single chip mode with no external RAM.

USING THE FFT PROGRAM

The FFT program along with test data to test the program is provided in the files FFT.MAC, TSTDAT.MAC and TWDTBL.MAC. TSTDAT.MAC contains the test data, and the output from the FFT routines. TWDTBL.MAC contains the Twiddle Factors. The FFT computation involves the use of 4 different subroutines: FFT, IFFT, BRNCNTR and SMULT. FFT does the forward FFT calculation, IFFT the Inverse FFT calculation, BRNCNTR implements the bit reversed counter, and SMULT does signed multiplication.

Two global symbols need to be defined by the user to use the FFT routines. The first, called TWSTAD should be set to the address of the start of the twiddle factor table. The second, called DTSTAD, should be set to the address of the start of the data area to be transformed. For details on the organization of these storage areas, see the preceding sections.

The actual number of data points to be transformed needs to be passed to the FFT routines. This is done as follows.

Two symbols that refer to words of on-chip RAM have been defined. The first is NUMB = W(01C0) and the second is L1 = W(01C2). Before calling the FFT routine, the user should load NUMB with N, the number of data points to be transformed, and L1 with L, $N = 2^L$.

To do a forward FFT, call FFT; to do an inverse FFT, call IFFT. In both cases, the output of the transform overwrites the input data.

INCREASING THE MAXIMUM TRANSFORM LENGTH

The maximum transform length for the FFT program is primarily limited by the size of the Twiddle Factor table. To increase the transform length, the following needs to be done.

1. Increase the Twiddle Factor table. Thus, if the maximum transform length required is 1024, the table needs to store the cosine and sine of the angles

   $0, 2\pi/1024, 2\pi \times 2/1024, \ldots, 2\pi \times 1023/1024$

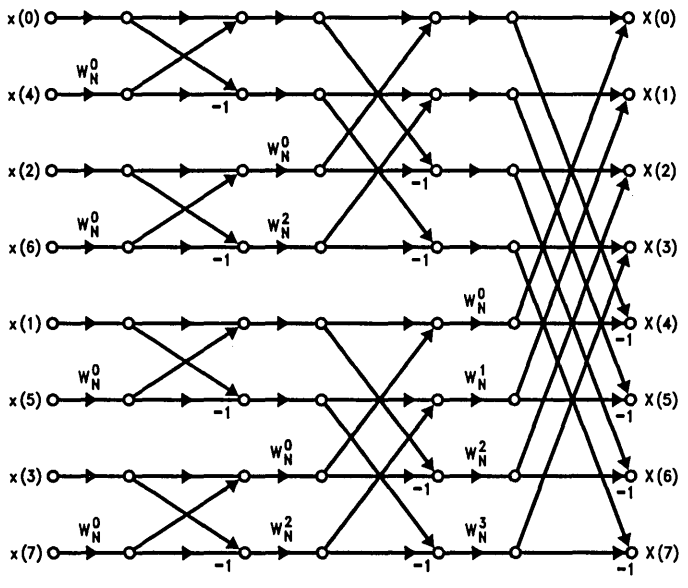
2. Change the global symbol LMAX such that the maximum transform length is 2^{LMAX}.

FFT/IFFT TEST PROGRAM

The data in the file TSTDAT.MAC can be used to test the FFT program. The data and its transform value is from reference 3. The program in reference 3 is for a Floating point FORTRAN FFT program. Since the HPC FFT program is a fixed point one, the input data needs to be suitably scaled. The scale factor chosen is 2^{10}. The file TSTDAT.MAC contains the scaled input data, and the expected transform. The input data is stored in memory words 200/27E and the expected transform is stored in memory words 280/2FE. To run the test program, do the following.

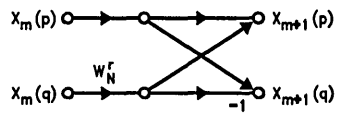
Set up the MOLE Development System with Blocks 0, 13, 14 and 15 mapped ON. Download the program to the MOLE. Set up a Breakpoint at F410. Run the program starting at F400. When the program is breakpointed, list memory words 200/27F and compare them with memory words 280/2FE.

Note that any difference between the expected DFT values and the DFT values actually computed is due to the fixed point computations in the FFT program.



TL/DD/9259–1

FIGURE 1. FFT Flow Graph for N = 8 Points



TL/DD/9259–2

FIGURE 2. The Butterfly—The Basic Computation Unit in the FFT

REFERENCES

1. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
2. L.R. Rabiner and B. Gold, *Theory and Applications of Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
3. IEEE ASSP Society Digital Signal Processing Committee, *Programs for Digital Signal Processing*, IEEE Press, New York, 1979.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

Listing of FFT Program Code

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


 1                      ;
 2                      ; THIS PROGRAM IMPLEMENTS A RADIX-2, DECIMATION IN TIME FFT ALGORITHM.
 3                      ;
 4                      ;
 5      0008                            LMAX = 08               ; MAXIMUM FFT LENGTH IS 2^LMAX.
 6                                                              ;
 7      F000                            TWSTAD = 0F000          ; TWIDDLE FACTOR TABLE START
 8                                                              ; ADDRESS.
 9      0200                            DTSTAD = 0200           ; DATA STORAGE AREA START
10                                                              ; ADDRESS.
11                      ;
12      01C0                            NUMB = W(01C0)          ; NUMBER OF DATA POINTS TO BE
13                                                              ; TRANSFORMED.
14      01C2                            L1 = W(01C2)            ; NUMB IS 2^L1.
15      01C4                            LSHIFT = W(01C4)        ; LSHIFT = LMAX - L1. IT IS A SHIFT
16                                                              ; FACTOR NEEDED TO COMPUTE THE
17                                                              ; ADDRESS REQUIRED FOR TWIDDLE
18                                                              ; FACTOR LOCKUP.
19      01C6                            NBFLY = W(01C6)         ; NUMBER OF BUTTERFLIES PER
20                                                              ; TWIDDLE FACTOR PER STAGE.
21      01C8                            ISTEP = W(01C8)         ; IF X(1) AND X(J) ARE INVOLVED
22                                                              ; IN A BUTTERFLY, THEN J=I+ISTEP.
23                                                              ; IT IS ALSO THE NUMBER OF TWIDDLE
24                                                              ; FACTORS IN A STAGE.
25      01CA                            ILEAP = W(01CA)         ; IF X(I) IS THE FIRST DATA VALUE
26                                                              ; FOR THE FIRST BUTTERFLY FOR A
27                                                              ; GIVEN TWIDDLE FACTOR, THEN THE
28                                                              ; SUBSEQUENT BUTTERFLIES FOR THAT
29                                                              ; TWIDDLE FACTOR HAVE AS THE FIRST
30                                                              ; DATA VALUE X(I+N*ILEAP).
31      01CC                            WESTEP = W(01CC)        ; TWIDDLE FACTOR EXPONENT STEP.
32                                                              ; THE TWIDDLE FACTORS FOR A GIVEN
33                                                              ; STAGE ARE W^(I*WESTEP).
34      01CE                            NSTG = W(01CE)          ; FFT STAGE BEING EVALUATED.
35                                                              ;
36      01D0                            ISTART = W(01D0)        ; INDEX OF THE FIRST DATA VALUE
37                                                              ; FOR THE FIRST BUTTERFLY FOR A
38                                                              ; GIVEN TWIDDLE FACTOR.
39      01D2                            WEXP = W(01D2)          ; EXPONENT VALUE FOR A GIVEN
40                                                              ; TWIDDLE FACTOR.
41      01D4                            NTWD = W(01D4)          ; TWIDDLE FACTOR BEING EVALUATED.
42                                                              ;
43      01D6                            COSTH = W(01D6)         ; COSINE PART OF TWIDDLE FACTOR.
44                                                              ;
45      01D8                            SINTH = W(01D8)         ; SINE PART OF TWIDDLE FACTOR.
46                                                              ;
47      01DA                            M1 = W(01DA)            ; INDEX OF FIRST DATA VALUE FOR
48                                                              ; A BUTTERFLY.
49      01DC                            NBCNT = W(01DC)         ; BUTTERFLY BEING EVALUATED.
50                                                              ;
51      01DE                            R1ADDR = W(01DE)        ; ADDRESS OF REAL PART OF FIRST
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     2
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


 52                                                          ; DATA VALUE INVOLVED IN A BUTTERFLY.
 53      01E0                          R2ADDR = W(01E0)      ; ADDRESS OF REAL PART OF SECOND
 54                                                          ; DATA VALUE INVOLVED IN A BUTTERFLY.
 55      01E2                          XR1 = W(01E2)         ; REAL PART OF FIRST DATA VALUE
 56                                                          ; INVOLVED IN A BUTTERFLY.
 57      01E4                          XI1= W(01E4)          ; IMAGINARY PART OF ABOVE.
 58                                                          ;
 59      01E6                          XR2 = W(01E6)         ; REAL PART OF SECOND DATA VALUE
 60                                                          ; INVOLVED IN A BUTTERFLY.
 61      01E8                          XI2 = W(01E8)         ; IMAGINARY PART OF ABOVE.
 62                                                          ;
 63      01EA                          TEMPR = W(01EA)       ; TEMPORARY STORAGE USED IN
 64                                                          ; A BUTTERFLY.
 65      01EC                          TEMPI = W(01EC)       ; SAME AS ABOVE.
 66                                                          ;
 67      01EE                          MTEMP = W(01EE)       ; TEMPORARY STORAGE USED IN SMULT.
 68                    ;
 69                                    .INCLD TSTDAT.MAC
 70                                    .INCLD TWDTBL.MAC
 71                    ;
 72                    ;
 73      F400                          . = 0F400
 74                    TSTFFT:
 75 F400 B701F0C4                     LD SP, 01F0
 76 F404 832001C0AB                   LD NUMB, 020          ; 32 POINT FFT.
 77 F409 830501C2AB                   LD L1, 05             ; 32 = 2^5.
 78 F40E 3049                         JSR FFT               ; COMPUTE FFT.
 79 F410 40                           NOP
 80 F411 31C4                         JSR IFFT
 81 F413 40                           NOP
 82 F414 61                           JP .-1
 83                    ;
 84                    ;
 85                    ; THIS SUBROUTINE IMPLEMENTS A BIT REVERSED COUNTER AS NEEDED FOR
 86                    ; DATA SHUFFLING IN THE FFT ROUTINE. THE ALGORITHM IS BASED ON
 87                    ; THE DESCRIPTION IN:
 88                    ;       RABINER AND GOLD,
 89                    ; THEORY AND APPLICATIONS OF DIGITAL SIGNAL PROCESSING,
 90                    ;       PRENTICE-HALL, 1975.
 91                    ;
 92                    ; ON INPUT, X CONTAINS THE PREVIOUS BIT REVERSED COUNTER VALUE.
 93                    ; THE NEXT BIT REVERSED OUTPUT IS RETURNED IN X.
 94                    ; A IS LOST, B AND K ARE PRESERVED.
 95                    ;
 96                                    .LOCAL
 97                    BRCNTR:
 98 F415 B601C0A8                     LD A, NUMB            ; GET NUMBER OF DATA SAMPLES
 99                                                          ; TO BE TRANSFORMED.
100 F419 C7                          SHR A                 ; DIVIDE BY 2.
101       $REPEAT:
102 F41A 96CEFD                      IFGT A, X             ; IS BIT BEING TESTED A 0 ?
```

5

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


103 F41D 47                             JP $FOUND               ; YES, SO STOP CHECKING.
104                                                             ; GET HERE MEANS BIT BEING
105                                                             ; CHECKED IS 1.
106 F41E 02                             SET C
107 F41F A0C8CEE8                       SUBC X, A               ; ZERO OUT THE BIT.
108 F423 C7                             SHR A                   ; UPDATE BIT LOCATOR.
109 F424 6A                             JP $REPEAT
110                         $FOUND:
111 F425 A0C8CEF8                       ADD X, A
112 F429 3C                             RET
113                                     .LOCAL
114                         ;
115                         ;
116                         ;
117                         ; THIS SUBROUTINE MULTIPLIES TWO 16-BIT 2'S COMPLEMENT INTEGERS AND RETURNS
118                         ; THE UPPER HALF OF THE RESULT. THE MULTIPLICAND IS IN A, AND THE MULTIPLIER
119                         ; IN W(B). THE RESULT IS RETURNED IN A. ONE TEMPORARY WORD OF STORAGE,
120                         ; ADDRESSED AS MTEMP IS USED.
121                         ;
122                         SMULT:
123 F42A 830001EEAB                     LD MTEMP, 0             ; CLEAR TEMPORARY STORAGE.
124 F42F A9CC                           INC B                   ; B NOW POINTS TO UPPER BYTE
125                                                             ; OF MULTIPLIER.
126 F431 17                             IF M(B).7               ; IS IT NEGATIVE ?
127 F432 B601EEAB                       ST A, MTEMP             ; THEN SAVE MULTIPLICAND IN MTEMP.
128 F436 AACC                           DECSZ B                 ; B INTO WORD POINTER.
129 F438 40                             NOP
130 F439 B601EEAE                       X A, MTEMP              ; SWAP A AND MTEMP.
131 F430 B601EF17                       IF M(($MTEMP)+1).7      ; IS MULTIPLICAND NEGATIVE ?
132 F441 F8                             ADD A, W(B)             ; THEN ACCUMULATE MULTIPLIER.
133 F442 B601EEAE                       X A, MTEMP
134 F446 FE                             MULT A, W(B)            ; UNSIGNED MULTIPLY.
135 F447 AECE                           X A, X                  ; UPPER HALF IN A.
136 F449 02                             SET C
137 F44A B601EEEB                       SUBC A, MTEMP
138 F44E E7                             SHL A
139 F44F 96CF17                         IF H(X).7
140 F452 04                             INC A
141 F453 E7                             SHL A
142 F454 96CF16                         IF H(X).6
143 F457 04                             INC A
144 F458 3C                             RET
145                         ;
146                         ;
147                         ;
148                         ; THIS SUBROUTINE IMPLEMENTS THE FIXED POINT RADIX-2 DECIMATION-IN-TIME
149                         ; FFT ALGORITHM. THE DATA IS INITIALLY PUT IN THE BIT REVERSED ORDER, AND
150                         ; THEN THE FFT IS COMPUTED. FOR THE THEORY BEHIND THE ALGORITHM, CONSULT:
151                         ;
152                         ;       1. OPPENHEIM AND SCHAFER, DIGITAL SIGNAL PROCESSING,
153                                         PRENTICE-HALL.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


154                     ;
155                     ;       2. RABINER AND GOLD, THEORY AND APPLICATIONS OF DIGITAL SIGNAL
156                     ;               PROCESSING, PRENTICE-HALL, 1975.
157                     ;
158                     ; THE ALGORITHM USED CLOSELY FOLLOWS THE FORTRAN PROGRAM FOREA IN
159                     ;
160                     ;       3. PROGRAMS FOR DIGITAL SIGNAL PROCESSING, IEEE.
161                     ;
162                     ;
163                     FFT:
164                     ;
165                     ; FIRST PUT THE DATA IN BIT REVERSED ORDER.
166                     ;
167 F459 00                             CLR A
168 F45A ABCC                          ST A, B                 ; SET UP NORMAL COUNTER.
169 F45C ABCE                          ST A, X                 ; SET UP BIT REVERSED COUNTER.
170 F45E A401C0CAAB                    LD K, NUMB              ; K HAS NUMBER OF DATA POINTS.
171                     REVLP:
172 F463 A0CCCEFD                      IFGT X, B               ; IS BIT REV CNTR --> NORM CNTR ?
173 F467 42                            JP SWAP                 ; YES, SO SWAP DATA.
174 F468 9421                          JMP COUNT               ; NO SO INCREMENT COUNT.
175                     SWAP:
176 F46A AFCC                          PUSH B
177 F46C AFCE                          PUSH X
178 F46E A8CC                          LD A, B                 ; INDEX VALUE I IS IN A.
179 F470 E7                            SHL A
180 F471 E7                            SHL A
181 F472 B80200                        ADD A, DTSTAD           ; GET ADDR. OF XR(I).
182 F475 ABCC                          ST A, B                 ; SAVE IT IN B.
183 F477 A8CE                          LD A, X                 ; INDEX VALUE J IS IN A.
184 F479 E7                            SHL A
185 F47A E7                            SHL A
186 F47B B80200                        ADD A, DISTAD           ; GET ADDR. OF XR(J).
187 F47E ABCE                          ST A, X                 ; SAVE IT IN X.
188 F480 E4                            LD A, W(B)              ; A <- XR(I).
189 F481 F1                            X A, W(X+)              ; A <- XR(J), XR(J) <- XR(I).
190 F482 E1                            XS A, W(B+)             ; A <- XR(I), XR(I) <- XR(J).
191 F483 40                            NOP
192 F484 E4                            LD A, W(B)              ; A <- XI(I).
193 F485 F5                            X A, W(X)               ; A <- XI(J), XI(J) <- XI(I).
194 F486 E6                            ST A, W(B)              ; XI(I) <- XI(J).
195 F487 3FCE                          POP X
196 F489 3FCC                          POP B
197                     ;
198                     COUNT:
199                     ;
200 F48B AACA                          DECSZ K                 ; DONE ?
201 F48D 41                            JP UPIT                 ; NO GO DO SOME MORE.
202 F48E 46                            JP DOFFT
203                     UPIT:
204 F48F 347A                          JSR BRCNTR              ; COUNT UP ON BIT REV CNTR.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


F491 A9CC                          INC B                ; COUNT UP ON NORMAL CNTR.
F493 9530                          JMP REVLP
                        DOFFT:
                        ;
                        ; DATA IS NOW STORED IN THE BIT REVERSED ORDER. COMPUTE THE FFT.
                        ;
F495 9008                          LD A, LMAX           ; A HAS MAX FFT EXPONENT.
F497 04                            INC A
F498 04                            INC A
F499 02                            SET C
F49A B601C2EB                      SUBC A, L1           ; COMPUTE LSHIFT.
F49E B601C4AB                      ST A, LSHIFT
F4A2 B601C0A8                      LD A, NUMB
F4A6 C7                            SHR A
F4A7 B601C6AB                      ST A, NBFLY          ; INITIALIZE NBFLY.
F4AB B601CCAB                      ST A, WESTEP         ; INITIALIZE WESTEP.
F4AF 830101C8AB                    LD ISTEP, 01         ; INITIALIZE ISTEP.
F4B4 830201CAAB                    LD ILEAP, 02         ; INITIALIZE ILEAP.
                        ;
                        ; SET UP L1 STAGES OF BUTTERFLIES.
                        ;
F4B9 B601C2AB                      LD A, L1
F4BD B601CEAB                      ST A, NSTG           ; LOOP L1 TIMES.
                        LOOP1:
                        ;
F4C1 00                            CLR A
F4C2 B601D0AB                      ST A, ISTART         ; INITIALIZE ISTART FOR EACH STAGE.
F4C6 B601D2AB                      ST A, WEXP           ; INITIALIZE WEXP.
                        ;
                        ; SET UP ISTEP LOOPS OF TWIDDLE FACTORS.
                        ;
F4CA B601C8A8                      LD A, ISTEP
F4CE B601D4AB                      ST A, NTWD           ; LOOP ISTEP TIMES.
                        LOOP2:
                        ;
                        ; LOOK UP THE TWIDDLE FACTOR.
                        ;
F4D2 A401C4CAAB                    LD K, LSHIFT         ; SHIFT LEFT LSHIFT TIMES.
F4D7 B601D2A8                      LD A, WEXP
                        GADLP:
F4DB E7                            SHL A
F4DC AACA                          DECSZ K              ; DONE SHIFTING ?
F4DE 63                            JP GADLP             ; NO SO DO MORE.
F4DF B8F000                        ADD A, TWSTAD        ; ADD STARTING ADDR OF TWIDDLE
                                                        ; FACTOR TABLE.
F4E2 ABCE                          ST A, X              ; TWIDDLE FACTOR ADDR IN X.
F4E4 F0                            LD A, W(X+)          ; GET COS(THETA).
F4E5 B601D6AB                      ST A, COSTH
F4E9 F4                            LD A, W(X)           ; GET SIN(THETA).
F4EA 01                            COMP A
F4EB 04                            INC A                ; MAKE IT NEGATIVE.
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


256 F4EC B601D8AB                   ST A, SINTH
257                       ;
258 F4F0 A501D001DAAB               LD M1, ISTART          ; INITIALIZE M1.
259                       ;
260                       ; SET UP NBFLY BUTTERFLIES FOR THIS TWIDDLE FACTOR.
261                       ;
262 F4F6 A501C601DCAB               LD NBCNT, NBFLY        ; LOOP NBFLY TIMES.
263                       LOOP3:
264 F4FC B601DAAB                  LD A, M1               ; GET INDEX OF X(I).
265 F500 E7                        SHL A
266 F501 E7                        SHL A
267 F502 B80200                    ADD A, DTSTAD          ; ADDR. OD XR(I).
268 F505 B601DEAB                  ST A, R1ADDR
269 F509 ABCE                      ST A, X
270 F50B F0                        LD A, W(X+)            ; A ← XR(I).
271 F50C B601E2AB                  ST A, XR1              ; STORE IN XR1.
272 F510 F4                        LD A, W(X)             ; A ← XI(I).
273 F511 B601E4AB                  ST A, XI1              ; STORE IN XI1.
274 F515 B601DAA8                  LD A, M1
275 F519 B601C8F8                  ADD A, ISTEP           ; GET INDEX OF X(J).
276 F51D E7                        SHL A
277 F51E E7                        SHL A
278 F51F B80200                    ADD A, DISTAD          ; ADDR. OF XR(J).
279 F522 B601E0AB                  ST A, R2ADDR
280 F526 ABCE                      ST A, X
281 F528 F0                        LD A, W(X+)            ; A ← XR(J).
282 F529 B601E6AB                  ST A, XR2              ; STORE IN XR2.
283 F520 F4                        LD A, W(X)             ; A ← XI(J).
284 F52E B601E8AB                  ST A, XI2              ; STORE IN XI2.
285                       ;
286 F532 B201E6                    LD B, #XR2             ; B ← ADDR(XR2).
287 F535 B601D6A8                  LD A, COSTH            ; A ← COS(THETA).
288 F539 350F                      JSR SMULT              ; COMPUTE XR(J)*COS(THETA).
289 F53B B601EAAB                  ST A, TEMPR            ; SAVE IN TEMPR.
290 F53F B601D8A8                  LD A, SINTH            ; A ← SIN(THETA).
291 F543 3519                      JSR SMULT              ; COMPUTE XR(J)*SIN(THETA).
292 F545 B601ECAB                  ST A, TEMPI            ; SAVE IN TEMPI.
293 F549 B201E8                    LD B, #XI2             ; B ← ADDR(XI2).
294 F54C B601D8A8                  LD A, SINTH            ; A ← SIN(THETA).
295 F550 3526                      JSR SMULT              ; COMPUTE XI(J)*SIN(THETA).
296 F552 01                        COMP A
297 F553 04                        INC A
298 F554 B601EAF8                  ADD A, TEMPR           ; COMPUTE XR(J)*COS(THETA) -
299                                                       ; XI(J)*SIN(THETA).
300 F558 B601EAAB                  ST A, TEMPR
301 F55C B601D6A8                  LD A, COSIN            ; A ← COS(THETA).
302 F560 3536                      JSR SMULT              ; COMPUTE XI(J)*COS(THETA).
303 F562 B601ECF8                  ADD A, TEMPI           ; COMPUTE XR(J)*SIN(THETA) +
304                                                       ; XI(J)*COS(THETA).
305 F566 B601ECAB                  ST A, TEMPI
306                       ;
```

```
                    ;
F56A A401DECEAB             LD X, R1ADDR          ; X ← ADDR(XR(I)).
F56F A401E0CCAB             LD B, R2ADDR          ; B ← ADDR(XR(J)).
F574 F0                     LD A, W(X+)           ; A ← XR(I).
F575 02                     SET C
F576 B601EAEB               SUBC A, TEMPR         ; A ← XR(I) - TEMPR.
F57A E1                     XS A, W(B+)
F57B 40                     NOP
F57C F2                     LD A, W(X-)           ; A ← XI(I).
F57D 02                     SET C
F57E B601ECEB               SUBC A, TEMPI         ; A ←XI(J) - TEMPI.
F582 E6                     ST A, W(B)
F583 F4                     LD A, W(X)            ; A ← XR(I).
F584 B601EAF8               ADD A, TEMPR          ; A ← XR(I) + TEMPR.
F588 F1                     X A, W(X+)
F589 F4                     LD A, W(X)            ; A ← XI(I).
F58A B601ECF8               ADD A, TEMPI          ; A ← XI(I) + TEMPI.
F58E F6                     ST A, W(X)
                    ;
F58F A501CA01DAF8           ADD M1, ILEAP         ; UPDATE M1 FOR NEXT LOOP.
                    ;
F595 B601DCAA               DECSZ NBCNT           ; DONE WITH ALL BUTTERFLIES
                                                  ; FOR THIS TWIDDLE FACTOR ?
F599 959D                   JMP LOOP3             ; NO, SO GO DO SOME MORE.
                    ;
                    ;
F59B B601D0A9               INC ISTART            ; SET UP STARTING INDEX FOR
                                                  ; NEXT TWIDDLE FACTOR.
F59F A501CC01D2F8           ADD WEXP, WESTEP      ; UPDATE TWIDDLE FACTOR
                                                  ; EXPONENT VALUE.
                    ;
F5A5 B601D4AA               DECSZ NTWD            ; DONE WITH ALL TWIDDLES
                                                  ; FOR THIS STAGE ?
F5A9 95D7                   JMP LOOP2             ; NO, SO GO DO SOME MORE.
                    ;
                    ;
F5AB B601CAA8               LD A, ILEAP
F5AF E7                     SHL A
F5B0 B601CAAB               ST A, ILEAP           ; UPDATE ILEAP FOR NEXT STAGE.
F5B4 B601C8A8               LD A, ISTEP
F5B8 E7                     SHL A
F5B9 B601C8AB               ST A, ISTEP           ; UPDATE ISTEP FOR NEXT STAGE.
F5BD B601C6A8               LD A, NBFLY
F5C1 C7                     SHR A
F5C2 B601C6AB               ST A, NBFLY           ; UPDATE NBFLY FOR NEXT STAGE.
F5C6 B601CCA8               LD A, WESTEP
F5CA C7                     SHR A
F5CB B601CCAB               ST A, WESTEP          ; UPDATE WESTEP FOR NEXT STAGE.
                    ;
F5CF B601CEAA               DECSZ NSTG            ; DONE WITH ALL STAGES ?
F5D3 B4FEEB       +         JMP LOOP1             ; NO SO GO DO SOME MORE.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     8
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


358                     ;
359 F5D6 3C                         RET                     ; ALL OVER.
360                     ;
361                     ; THE CODE BELOW IS FOR THE INVERSE FFT. THE ONLY DIFFERENCE IS THAT
362                     ; THE TWIDDLE FACTORS ARE USED A LITTLE DIFFERENTLY, AND A FINAL SCALING BY
363                     ; 1/NUMB IS DONE.
364                     IFFT:
365                     ;
366                     ; FIRST PUT THE DATA IN BIT REVERSED ORDER.
367                     ;
368 F5D7 00                          CLR A
369 F5D8 ABCC                       ST A, B                 ; SET UP NORMAL COUNTER.
370 F5DA ABCE                       ST A, X                 ; SET UP BIT REVERSED COUNTER.
371 F5DC A401C0CAAB                 LD K, NUMB              ; K HAS NUMBER OF DATA POINTS.
372                     IREVLP:
373 F5E1 A0CCCEFD                   IFGT X, B               ; IS BIT REV CNTR --> NORM CNTR ?
374 F5E5 42                         JP ISWAP                ; YES, SO SWAP DATA.
375 F5E6 9421                       JMP ICOUNT              ; NO SO INCREMENT COUNT.
376                     ISWAP:
377 F5E8 AFCC                       PUSH B
378 F5EA AFCE                       PUSH X
379 F5EC A8CC                       LD A, B                 ; INDEX VALUE I IS IN A.
380 F5EE E7                         SHL A
381 F5EF E7                         SHL A
382 F5F0 B80200                     ADD A, DTSTAD           ; GET ADDR. OF XR(I).
383 F5F3 ABCC                       ST A, B                 ; SAVE IT IN B.
384 F5F5 A8CE                       LD A, X                 ; INDEX VALUE J IS IN A.
385 F5F7 E7                         SHL A
386 F5F8 E7                         SHL A
387 F5F9 B80200                     ADD A, DTSTAD           ; GET ADDR. OF XR(J).
388 F5FC ABCE                       ST A, X                 ; SAVE IT IN X.
389 F5FE E4                         LD A, W(B)              ; A <- XR(I).
390 F5FF F1                         X A, W(X+)              ; A <- XR(J), XR(J) <- XR(I).
391 F600 E1                         XS A, W(B+)             ; A <- XR(I), XR(I) <- XR(J).
392 F601 40                         NOP
393 F602 E4                         LD A, W(B)              ; A <- XI(I).
394 F603 F5                         X A, W(X)               ; A <- XI(J), XI(J) <- XI(I).
395 F604 E6                         ST A, W(B)              ; XI(I) <- XI(J).
396 F605 3FCE                       POP X
397 F607 3FCC                       POP B
398                     ;
399                     ICOUNT:
400                     ;
401 F609 AACA                       DECSZ K                 ; DONE ?
402 F60B 41                         JP IUPIT                ; NO GO DO SOME MORE.
403 F60C 46                         JP DOIFFT
404                     IUPIT:
405 F60D 35F8                       JSR BRCNTR              ; COUNT UP ON BIT REV CNTR.
406 F60F A9CC                       INC B                   ; COUNT UP ON NORMAL CNTR.
407 F611 9530                       JMP IREVLP
408                     DOIFFT:
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     9
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


409                     ;
410                     ; DATA IS NOW STORED IN THE BIT REVERSED ORDER. COMPUTE THE FFT.
411                     ;
412 F613 9008                          LD A, LMAX              ; A HAS MAX FFT EXPONENT.
413 F615 04                            INC A
414 F616 04                            INC A
415 F617 02                            SET C
416 F618 B601C2EB                      SUBC A, L1              ; COMPUTE LSHIFT.
417 F61C B601C4AB                      ST A, LSHIFT
418 F620 B601C0A8                      LD A, NUMB
419 F624 C7                            SHR A
420 F625 B601C6AB                      ST A, NBFLY             ; INITIALIZE NBFLY.
421 F629 B601CCAB                      ST A, WESTEP            ; INITIALIZE WESTEP.
422 F62D 830101C8AB                    LD ISTEP, 01            ; INITIALIZE ISTEP.
423 F632 830201CAAB                    LD ILEAP, 02            ; INITIALIZE ILEAP.
424                     ;
425                     ; SET UP L1 STAGES OF BUTTERFLIES.
426                     ;
427 F637 B601C2AB                      LD A, L1
428 F63B B601CEAB                      ST A, NSTG              ; LOOP L1 TIMES.
429                     ILOOP1:
430                     ;
431 F63F 00                            CLR A
432 F640 B601D0AB                      ST A, ISTART            ; INITIALIZE ISTART FOR EACH STAGE.
433 F644 B601D2AB                      ST A, WEXP              ; INITIALIZE WEXP.
434                     ;
435                     ; SET UP ISTEP LOOPS OF TWIDDLE FACTORS.
436                     ;
437 F648 B601C8A8                      LD A, ISTEP
438 F64C B601D4AB                      ST A, NTWD              ; LOOP ISTEP TIMES.
439                     ILOOP2:
440                     ;
441                     ; LOOK UP THE TWIDDLE FACTOR.
442                     ;
443 F650 A401C4CAAB                    LD K, LSHIFT            ; SHIFT LEFT LSHIFT TIMES.
444 F655 B601D2AB                      LD A, WEXP
445                     IGADLP:
446 F659 E7                            SHL A
447 F65A AACA                          DECSZ K                 ; DONE SHIFTING ?
448 F65C 63                            JP IGADLP               ; NO DO SOME MORE.
449 F65D B8F000                        ADD A, TWSTAD           ; ADD STARTING ADDR OF TWIDDLE
450                                                            ; FACTOR TABLE.
451 F660 ABCE                          ST A, X                 ; TWIDDLE FACTOR ADDR IN X.
452 F662 F0                            LD A, W(X+)             ; GET COS(THETA).
453 F663 B601D6AB                      ST A, COSTH
454 F667 F4                            LD A, W(X)              ; GET SIN(THETA).
455 F668 B601D8AB                      ST A, SINTH
456                     ;
457 F66C A501D001DAAB                  LD M1, ISTART           ; INITIALIZE M1.
458                     ;
459                     ; SET UP NBFLY BUTTERFLIES FOR THIS TWIDDLE FACTOR.
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     10
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


460                     ;
461 F672 A501C601DCAB                   LD NBCNT, NBFLY         ; LOOP NBFLY TIMES.
462                     ILOOP3:
463 F678 B601DAA8                       LD A, M1                ; GET INDEX OF X(I).
464 F67C E7                             SHL A
465 F67D E7                             SHL A
466 F67E B80200                         ADD A, DTSTAD           ; ADDR. OD XR(I).
467 F681 B601DEA8                       ST A, R1ADDR
468 F685 ABCE                           ST A, X
469 F687 F0                             LD A, W(X+)             ; A ← XR(I).
470 F688 B601E2AB                       ST A, XR1               ; STORE IN XR1.
471 F68C F4                             LD A, W(X)              ; A ← XI(I).
472 F68D B601E4AB                       ST A, XI1               ; STORE IN XI1.
473 F691 B601DAA8                       LD A, M1
474 F695 B601C8F8                       ADD A, ISTEP            ; GET INDEX OF X(J).
475 F699 E7                             SHL A
476 F69A E7                             SHL A
477 F69B B80200                         ADD A, DTSTAD           ; ADDR. OF XR(J).
478 F69E B601E0AB                       ST A, R2ADDR
479 F6A2 ABCE                           ST A, X
480 F6A4 F0                             LD A, W(X+)             ; A ← XR(J).
481 F6A5 B601E6AB                       ST A, XR2               ; STORE IN XR2.
482 F6A9 F4                             LD A, W(X)              ; A ← XI(J).
483 F6AA B601E8AB                       ST A, XI2               ; STORE IN XI2.
484                     ;
485 F6A8 B201E6                         LD B, #XR2              ; B ← ADDR(XR2).
486 F6B1 B601D6A8                       LD A, COSTH             ; A ← COS(THETA).
487 F6B5 368B                           JSR SMULT               ; COMPUTE XR(J)*COS(THETA).
488 F6B7 B601EAAB                       ST A, TEMPR             ; SAVE IN TEMPR.
489 F6BB B601D8A8                       LD A, SINTH             ; A ← SIN(THETA).
490 F6BF 3695                           JSR SMULT               ; COMPUTE XR(J)*SIN(THETA).
491 F6C1 B601ECAB                       ST A, TEMPI             ; SAVE IN TEMPI.
492 F6C5 B201E8                         LD B, #XI2              ; B ← ADDR(XI2).
493 F6C8 B601D8A8                       LD A, SINTH             ; A ← SIN(THETA).
494 F6CC 36A2                           JSR SMULT               ; COMPUTE XI(J)*SIN(THETA).
495 F6CE 01                             COMP A
496 F6CF 04                             INC A
497 F6D0 B601EAF8                       ADD A, TEMPR            ; COMPUTE XR(J)*COS(THETA) -
498                                                             ; XI(J)*SIN(THETA).
499 F6D4 B601EAAB                       ST A, TEMPR
500 F6D8 B601D6A8                       LD A, COSTH             ; A ← COS(THETA).
501 F6DC 36B2                           JSR SMULT               ; COMPUTE XI(J)*COS(THETA).
502 F6DE B601ECF8                       ADD A, TEMPI            ; COMPUTE XR(J)*SIN(THETA) +
503                                                             ; XI(J)*COS(THETA).
504 F6E2 B601ECAB                       ST A, TEMPI
505                     ;
506                     ;
507 F6E6 A401DECEAB                     LD X, R1ADDR            ; X ← ADDR(XR(I)).
508 F6EB A401E0CCAB                     LD B, R2ADDR            ; B ← ADDR(XR(J)).
509 F6F0 F0                             LD A, W(X+)             ; A ← XR(I).
510 F6F1 02                             SET C
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     11
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



511 F6F2 B601EAEB                   SUBC A, TEMPR           ; A ← XR(I) - TEMPR.
512 F6F6 E1                         XS A, W(B+)
513 F6F7 40                         NOP
514 F6F8 F2                         LD A, W(X-)             ; A ← XI(I).
515 F6F9 02                         SET C
516 F6FA B601ECEB                   SUBC A, TEMPI           ; A ← XI(J) - TEMPI.
517 F6FE E6                         ST A, W(B)
518 F6FF F4                         LD A, W(X)              ; A ← XR(I).
519 F700 B601EAF8                   ADD A, TEMPR            ; A ← XR(I) + TEMPR.
520 F704 F1                         X A, W(X+)
521 F705 F4                         LD A, W(X)              ; A ← XI(I).
522 F706 B601ECF8                   ADD A, TEMPI            ; A ← XI(I) + TEMPI.
523 F70A F6                         ST A, W(X)
524                     ;
525 F70B A501CA01DAF8               ADD M1, ILEAP           ; UPDATE M1 FOR NEXT LOOP.
526                     ;
527 F711 B601DCAA                   DECSZ NBCNT             ; DONE WITH ALL BUTTERFLIES
528                                                         ; FOR THIS TWIDDLE FACTOR ?
529 F715 959D                       JMP ILOOP3              ; NO, SO GO DO SOME MORE.
530                     ;
531                     ;
532 F717 B601D0A9                   INC ISTART              ; SET UP STARTING INDEX FOR
533                                                         ; NEXT TWIDDLE FACTOR.
534 F71B A501CC01D2F8               ADD WEXP, WESTEP        ; UPDATE TWIDDLE FACTOR
535                                                         ; EXPONENT VALUE.
536                     ;
537 F721 B601D4AA                   DECSZ NTWD              ; DONE WITH ALL TWIDDLES
538                                                         ; FOR THIS STAGE ?
539 F725 95D5                       JMP ILOOP2              ; NO, SO GO DO SOME MORE.
540                     ;
541                     ;
542 F727 B601CAA8                   LD A, ILEAP
543 F72B E7                         SHL A
544 F72C B601CAAB                   ST A, ILEAP             ; UPDATE ILEAP FOR NEXT STAGE.
545 F730 B601C8A8                   LD A, ISTEP
546 F734 E7                         SHL A
547 F735 B601C8AB                   ST A, ISTEP             ; UPDATE ISTEP FOR NEXT STAGE.
548 F739 B601C6A8                   LD A, NBFLY
549 F73D C7                         SHR A
550 F73E B601C6AB                   ST A, NBFLY             ; UPDATE NBFLY FOR NEXT STAGE.
551 F742 B601CCA8                   LD A, WESTEP
552 F746 C7                         SHR A
553 F747 B601CCAB                   ST A, WESTEP            ; UPDATE WESTEP FOR NEXT STAGE.
554                     ;
555 F748 B601CEAA                   DECSZ NSTG              ; DONE WITH ALL STAGES ?
556 F74F B4FEED       +             JMP ILOOP1              ; NO SO GO DO SOME MORE.
557                     ;
558                     ;
559                     ; DO THE FINAL SCALING OF THE DATA BY 1/NUMB.
560                     ;
561 F752 B04000                     LD A, 04000             ; A ← 1.0
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     12
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


562 F755 A401C2CAAB                  LD K, L1              ; K ← L1.
563                      SCALLP:
564 F75A C7                          SHR A                 ; DIVIDE BY 2.
565 F75B AACA                        DECSZ K
566 F75D 63                          JP SCALLP
567                                                        ;
568                                                        ; GET HERE MEANS A IS 1/(2*L1).
569 F75E B601EAAB                    ST A, TEMPR           ; SAVE IT IN TEMPR.
570 F762 A501C001DCAB                LD NBCNT, NUMB        ; LOOP COUNTER.
571 F768 B20200                      LD B, DTSTAD          ; B ← ADDR(XR(0)).
572                      SCALIT:
573 F76B B601EAA8                    LD A, TEMPR
574 F76F 3745                        JSR SMULT             ; A ← XR(I)*(1/NUMB).
575 F771 E1                          XS A, W(B+)           ; XR(I) ← XR(I)*(1/NUMB).
576 F772 40                          NOP
577 F773 B601EAA8                    LD A, TEMPR           ; A ← 1/NUMB.
578 F777 374D                        JSR SMULT             ; A ← X1(I)*(1/NUMB).
579 F779 E1                          XS A, W(B+)           ; XI(I) ← XI(I)*(1/NUMB).
580 F77A 40                          NOP
581 F77B B601DCAA                    DECSZ NBCNT           ; DONE ?
582 F77F 74                          JP SCALIT             ; NO DO SOME MORE.
583 F780 3C                          RET                   ; ALL OVER.
584                      ;
585 FFFE 00F4                        .END TSTFFT
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:     13
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

SYMBOL TABLE


A       00C8 W      B       00CC W      BRCNTR F415         COSTN  01D6 W
COUNT   F48B        DOFFT   F495        DOIFFT F613         DTSTAD 0200
FFT     F459        GADLP   F40B        ICOUNT F609         IFFT   F507
IGADLP F659         ILEAP   01CA W      IL00P1 F63F         IL00P2 F650
IL00P3 F678         IREVLP F5E1         ISTART 01D0 W       ISTEP  01C8 W
ISWAP   F5E8        IUPIT   F600        K       00CA W      L1     01C2 W
LMAX    0008        L00P1   F4C1        L00P2   F4D2        L00P3  F4FC
LSHIFT 01C4 W       M1      01DA W      MTEMP   01EE W      NBCNT  01DC W
NBFLY   01C6 W      NSTG    01CE W      NTWD    01D4 W      NUMB   01C0 W
PC      00C6 W      R1ADDR  01DE W      R2ADDR  01E0 W      REVLP  F463
SCALIT F76B         SCALLP  F75A        SINTH   01D8 W      SMULT  F42A
SP      00C4 W      SWAP    F46A        TEMPI   01EC W      TEMPR  01EA W
TSTFFT F400         TWSTAD  F000        UPIT    F48F        WESTEP 01CC W
WEXP    01D2 W      X       00CE W      XI1     01E4 W      XI2    01E8 W
XR1     01E2 W      XR2     01E6 W      $FOUND  F425        $REPEA F41A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     14
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

MACRO TABLE


   NO WARNING LINES

   NO ERROR LINES

 2307  ROM BYTES USED

SOURCE CHECKSUM = E9FC
OBJECT CHECKSUM = 28FC

INPUT   FILE C:FFT.MAC
LISTING FILE C:FFT.PRN
OBJECT  FILE C:FFT.LM
```

APPENDIX B

Twiddle Factor Table

```
;
; TWIDDLE FACTOR TABLE FOR USE IN THE FFT ROUTINES.
;
; TABLE SET FOR MAX FFT LENGTH OF 256.
;
; TABLE STARTS AT F000 AND OCCUPIES 1024 BYTES OF STORAGE.
;
                . = 0F000
                .WORD 16384, 0
                .WORD 16379, 402
                .WORD 16364, 804
                .WORD 16340, 1205
                .WORD 16305, 1606
                .WORD 16261, 2006
                .WORD 16207, 2404
                .WORD 16143, 2801
                .WORD 16069, 3196
                .WORD 15986, 3590
                .WORD 15893, 3981
                .WORD 15791, 4370
                .WORD 15679, 4756
                .WORD 15557, 5139
                .WORD 15426, 5520
                .WORD 15286, 5897
                .WORD 15137, 6270
                .WORD 14978, 6639
                .WORD 14811, 7005
                .WORD 14635, 7366
                .WORD 14449, 7723
                .WORD 14256, 8076
                .WORD 14053, 8423
                .WORD 13842, 8765
                .WORD 13623, 9102
                .WORD 13395, 9434
                .WORD 13160, 9760
                .WORD 12916, 10080
                .WORD 12665, 10394
                .WORD 12406, 10702
                .WORD 12140, 11003
                .WORD 11866, 11297
                .WORD 11585, 11585
                .WORD 11297, 11866
                .WORD 11003, 12140
                .WORD 10702, 12406
                .WORD 10394, 12665
                .WORD 10080, 12916
                .WORD 9760, 13160
                .WORD 9434, 13395
                .WORD 9102, 13623
                .WORD 8765, 13842
                .WORD 8423, 14053
                .WORD 8076, 14256
                .WORD 7723, 14449
                .WORD 7366, 14635
                .WORD 7005, 14811
                .WORD 6639, 14978
                .WORD 6270, 15137
                .WORD 5897, 15286
                .WORD 5520, 15426
                .WORD 5139, 15557
                .WORD 4756, 15679
                .WORD 4370, 15791
                .WORD 3981, 15893
                .WORD 3590, 15986
                .WORD 3196, 16069
                .WORD 2801, 16143
                .WORD 2404, 16207
                .WORD 2006, 16261
                .WORD 1606, 16305
                .WORD 1205, 16340
                .WORD 804, 16364
                .WORD 402, 16379
                .WORD 0, 16384
                .WORD -402, 16379
                .WORD -804, 16364
                .WORD -1205, 16340
                .WORD -1606, 16305
                .WORD -2006, 16261
                .WORD -2404, 16207
                .WORD -2801, 16143
                .WORD -3196, 16069
                .WORD -3590, 15986
                .WORD -3981, 15893
                .WORD -4370, 15791
                .WORD -4756, 15679
                .WORD -5139, 15557
                .WORD -5520, 15426
                .WORD -5897, 15286
                .WORD -6270, 15137
                .WORD -6639, 14978
                .WORD -7005, 14811
                .WORD -7366, 14635
                .WORD -7723, 14449
                .WORD -8076, 14256
                .WORD -8423, 14053
                .WORD -8765, 13842
                .WORD -9102, 13623
                .WORD -9434, 13395
                .WORD -9760, 13160
                .WORD -10080, 12916
                .WORD -10394, 12665
                .WORD -10702, 12406
                .WORD -11003, 12140
                .WORD -11297, 11866
                .WORD -11585, 11585
                .WORD -11866, 11297
                .WORD -12140, 11003
                .WORD -12406, 10702
                .WORD -12665, 10394
                .WORD -12916, 10080
```

```
.WORD -13160, 9760
.WORD -13395, 9434
.WORD -13623, 9102
.WORD -13842, 8765
.WORD -14053, 8423
.WORD -14256, 8076
.WORD -14449, 7723
.WORD -14635, 7366
.WORD -14811, 7005
.WORD -14978, 6639
.WORD -15137, 6270
.WORD -15286, 5897
.WORD -15426, 5520
.WORD -15557, 5139
.WORD -15679, 4756
.WORD -15791, 4370
.WORD -15893, 3981
.WORD -15986, 3590
.WORD -16069, 3196
.WORD -16143, 2801
.WORD -16207, 2404
.WORD -16261, 2006
.WORD -16305, 1606
.WORD -16340, 1205
.WORD -16364, 804
.WORD -16379, 402
.WORD -16384, 0

.WORD -16379, -402
.WORD -16364, -804
.WORD -16340, -1205
.WORD -16305, -1606
.WORD -16261, -2006
.WORD -16207, -2404
.WORD -16143, -2801
.WORD -16069, -3196
.WORD -15986, -3590
.WORD -15893, -3981
.WORD -15791, -4370
.WORD -15679, -4756
.WORD -15557, -5139
.WORD -15426, -5520
.WORD -15286, -5897
.WORD -15137, -6270
.WORD -14978, -6639
.WORD -14811, -7005
.WORD -14635, -7366
.WORD -14449, -7723
.WORD -14256, -8076
.WORD -14053, -8423
.WORD -13842, -8765
.WORD -13623, -9102
.WORD -13395, -9434
.WORD -13160, -9760

.WORD -12916, -10080
.WORD -12665, -10394
.WORD -12406, -10702
.WORD -12140, -11003
.WORD -11866, -11297
.WORD -11585, -11585
.WORD -11297, -11866
.WORD -11003, -12140
.WORD -10702, -12406
.WORD -10394, -12665
.WORD -10080, -12916
.WORD -9760, -13160
.WORD -9434, -13395
.WORD -9102, -13623
.WORD -8765, -13842
.WORD -8423, -14053
.WORD -8076, -14256
.WORD -7723, -14449
.WORD -7366, -14635
.WORD -7005, -14811
.WORD -6639, -14978
.WORD -6270, -15137
.WORD -5897, -15286
.WORD -5520, -15426
.WORD -5139, -15557
.WORD -4756, -15679
.WORD -4370, -15791
.WORD -3981, -15893
.WORD -3590, -15986
.WORD -3196, -16069
.WORD -2801, -16143
.WORD -2404, -16207
.WORD -2006, -16261
.WORD -1606, -16305
.WORD -1205, -16340
.WORD -804, -16364
.WORD -402, -16379
.WORD 0, -16384
.WORD 402, -16379
.WORD 804, -16364
.WORD 1205, -16340
.WORD 1606, -16305
.WORD 2006, -16261
.WORD 2404, -16207
.WORD 2801, -16143
.WORD 3196, -16069
.WORD 3590, -15986
.WORD 3981, -15893
.WORD 4370, -15791
.WORD 4756, -15679
.WORD 5139, -15557
.WORD 5520, -15426
.WORD 5897, -15286
.WORD 6270, -15137
.WORD 6639, -14978
```

```
.WORD 7005, -14811
.WORD 7366, -14635
.WORD 7723, -14449
.WORD 8076, -14256
.WORD 8423, -14053
.WORD 8765, -13842
.WORD 9102, -13623
.WORD 9434, -13395
.WORD 9760, -13160
.WORD 10080, -12916
.WORD 10394, -12665
.WORD 10702, -12406
.WORD 11003, -12140
.WORD 11297, -11866
.WORD 11585, -11585
.WORD 11866, -11297
.WORD 12140, -11003
.WORD 12406, -10702
.WORD 12665, -10394
.WORD 12916, -10080
.WORD 13160, -9760
.WORD 13395, -9434
.WORD 13623, -9102
.WORD 13842, -8765
.WORD 14053, -8423
.WORD 14256, -8076
.WORD 14449, -7723
.WORD 14635, -7366
.WORD 14811, -7005
.WORD 14978, -6639
.WORD 15137, -6270
.WORD 15286, -5897
.WORD 15426, -5520
.WORD 15557, -5139
.WORD 15679, -4756
.WORD 15791, -4370
.WORD 15893, -3981
.WORD 15986, -3590
.WORD 16069, -3196
.WORD 16143, -2801
.WORD 16207, -2404
.WORD 16261, -2006
.WORD 16305, -1606
.WORD 16340, -1205
.WORD 16364, -804
.WORD 16379, -402

.END
```

APPENDIX C

Test Data and Expected Results

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


  1                     ;
  2                     ;
  3                     ; TEST DATA FOR FFT ROUTINES.
  4                     ; OBTAINED FROM : PROGRAMS FOR DIGITAL SIGNAL PROCESSING, IEEE PRESS,
  5                     ; CHAPTER 1 BY GOLD.
  6                     ;
  7                     ;
  8       0200                          . = 0200
  9 0200 0004                         .WORD 1024, 0
    0202 0000
 10 0204 9A03                         .WORD 922, 307
    0206 3301
 11 0208 E102                         .WORD 737, 553
    020A 2902
 12 020C F201                         .WORD 498, 719
    020E CF02
 13 0210 E800                         .WORD 232, 796
    0212 1C03
 14 0214 E2FF                         .WORD -30, 786
    0216 1203
 15 0218 F9FE                         .WORD -263, 699
    021A BB02
 16 021C 42FE                         .WORD -446, 550
    021E 2602
 17 0220 C9FD                         .WORD -567, 361
    0222 6901
 18 0224 96FD                         .WORD -618, 155
    0226 9800
 19 0228 A5FD                         .WORD -603, -46
    022A D2FF
 20 022C EFFD                         .WORD -529, -222
    022E 22FF
 21 0230 67FE                         .WORD -409, -359
    0232 99FE
 22 0234 FBFE                         .WORD -261, -446
    0236 42FE
 23 0238 9BFF                         .WORD -101, -479
    023A 21FE
 24 023C 3500                         .WORD 53, -462
    023E 32FE
 25 0240 BA00                         .WORD 186, -400
    0242 70FE
 26 0244 1F01                         .WORD 287, -304
    0246 D0FE
 27 0248 5E01                         .WORD 350, -187
    024A 45FF
 28 024C 7301                         .WORD 371, -64
    024E C0FF
 29 0250 6101                         .WORD 353, 54
    0252 3600
 30 0254 2001                         .WORD 301, 154
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     2
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


   0256 9A00
31 0258 E100                          .WORD 225, 229
   025A E500
32 025C 8600                          .WORD 134, 274
   025E 1201
33 0260 2600                          .WORD 38, 287
   0262 1F01
34 0264 CCFF                          .WORD -52, 269
   0266 0D01
35 0268 81FF                          .WORD -127, 227
   026A E300
36 026C 49FF                          .WORD -183, 166
   026E A600
37 0270 2AFF                          .WORD -214, 95
   0272 5F00
38 0274 23FF                          .WORD -221, 21
   0276 1500
39 0278 33FF                          .WORD -205, -48
   027A DDFF
40 027C 55FF                          .WORD -171, -104
   027E 98FF
41                      ;
42                      ; THESE ARE THE EXPECTED DFT RESULTS.
43                      ;
44 0280 C702                          .WORD 711, 3584
   0282 000E
45 0284 2B0B                          .WORD 2859, 8244
   0286 3420
46 0288 9D25                          .WORD 9629, -9354
   028A 76DB
47 028C 7707                          .WORD 1911, -3926
   028E AAF0
48 0290 8704                          .WORD 1159, -2288
   0292 10F7
49 0294 9F03                          .WORD 927, -1571
   0296 DDF9
50 0298 3303                          .WORD 819, -1167
   029A 71FB
51 029C F502                          .WORD 757, -903
   029E 79FC
52 02A0 CE02                          .WORD 718, -715
   02A2 35FD
53 02A4 B202                          .WORD 690, -572
   02A6 C4FD
54 02A8 9D02                          .WORD 669, -457
   02AA 37FE
55 02AC 8C02                          .WORD 652, -361
   02AE 97FE
56 02B0 7F02                          .WORD 639, -279
   02B2 E9FE
57 02B4 7302                          .WORD 627, -205
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



   0286 33FF
58 02B8 6902                          .WORD 617, -139
   02BA 75FF
59 02BC 6002                          .WORD 608, -77
   02BE B3FF
60 02C0 5802                          .WORD 600, -18
   02C2 EEFF
61 02C4 5102                          .WORD 593, 39
   02C6 2700
62 02C8 4A02                          .WORD 586, 95
   02CA 5F00
63 02CC 4302                          .WORD 579, 152
   02CE 9800
64 02D0 3C02                          .WORD 572, 210
   02D2 0200
65 02D4 3502                          .WORD 565, 271
   02D6 0F01
66 02D8 2E02                          .WORD 558, 336
   02DA 5001
67 02DC 2702                          .WORD 551, 408
   02DE 9801
68 02E0 2002                          .WORD 544, 488
   02E2 EB01
69 02E4 1802                          .WORD 536, 581
   02E6 4502
70 02E8 1002                          .WORD 528, 691
   02EA B302
71 02EC 0702                          .WORD 519, 827
   02EE 3B03
72 02F0 FE01                          .WORD 510, 1004
   02F2 EC03
73 02F4 F701                          .WORD 503, 1248
   02F6 E004
74 02F8 F701                          .WORD 503, 1615
   02FA 4F06
75 02FC 1202                          .WORD 530, 2241
   02FE C108
76                      ;
77                      ;
78                      ; TEST DATA FOR FFT ROUTINES.
79                      ; OBTAINED FROM :
80                      ;
81                      ;
82 0300 0004                          .WORD 1024, 0
   0302 0000
83 0304 9A03                          .WORD 922, 307
   0306 3301
84 0308 E102                          .WORD 737, 553
   030A 2902
85 030C F201                          .WORD 498, 719
   030E CF02
```

```
 86 0310 E800                       .WORD 232, 796
    0312 1C03
 87 0314 E2FF                       .WORD -30, 786
    0316 1203
 88 0318 F9FE                       .WORD -263, 699
    031A BB02
 89 031C 42FE                       .WORD -446, 550
    031E 2602
 90 0320 C9FD                       .WORD -567, 361
    0322 6901
 91 0324 96FD                       .WORD -618, 155
    0326 9B00
 92 0328 A5FD                       .WORD -603, -46
    032A D2FF
 93 032C EFFD                       .WORD -529, -222
    032E 22FF
 94 0330 67FE                       .WORD -409, -359
    0332 99FE
 95 0334 FBFE                       .WORD -261, -446
    0336 42FE
 96 0338 9BFF                       .WORD -101, -479
    033A 21FE
 97 033C 3500                       .WORD 53, -462
    033E 32FE
 98 0340 BA00                       .WORD 186, -400
    0342 70FE
 99 0344 1F01                       .WORD 287, -304
    0346 D0FE
100 0348 5E01                       .WORD 350, -187
    034A 45FF
101 034C 7301                       .WORD 371, -64
    034E C0FF
102 0350 6101                       .WORD 353, 54
    0352 3600
103 0354 2D01                       .WORD 301, 154
    0356 9A00
104 0358 E100                       .WORD 225, 229
    035A E500
105 035C 8600                       .WORD 134, 274
    035E 1201
106 0360 2600                       .WORD 38, 287
    0362 1F01
107 0364 CCFF                       .WORD -52, 269
    0366 0D01
108 0368 81FF                       .WORD -127, 227
    036A E300
109 036C 49FF                       .WORD -183, 166
    036E A600
110 0370 2AFF                       .WORD -214, 95
    0372 5F00
111 0374 23FF                       .WORD -221, 21
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



    0376 1500
112 0378 33FF                           .WORD -205, -48
    037A DOFF
113 037C 55FF                           .WORD -171, -104
    037E 98FF
114                     ;
115                                     .END
                                          @
ERROR, OPERAND MUST BE SINGLE VALID SYMBOL NAME



NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

SYMBOL TABLE


A       00C8 W      B       00CC W      K       00CA W      PC      00C6 W
SP      00C4 W      X       00CE W
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

MACRO TABLE



   NO WARNING LINES

    1 ERROR LINES

  384  ROM BYTES USED

SOURCE CHECKSUM = 7A03
OBJECT CHECKSUM = 0705

INPUT   FILE C:TSTDAT.MAC
LISTING FILE C:TSTDAT.PRN
```

Expanding the HPC Address Space

National Semiconductor
Application Note 497
Joe Cocovich

INTRODUCTION

The maximum address range of the HPC family of 16-bit High Performance microControllers is 64k bytes using the external address/data bus to interface with external memory. This application note describes a method to increase the amount of memory in a system to 544k bytes utilizing bank switching techniques. Block diagrams are presented to aid in circuit design. Software examples are given for memory and bank management.

HPC ADDRESSING

Program memory addressing is accomplished by the 16-bit Program Counter on a byte basis (instructions are always fetched a byte at a time). Memory can be addressed as words or bytes directly by instructions or indirectly through the B, X and SP registers. Words are always addressed on even-byte boundaries. The HPC uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The external address/data bus of the HPC is 16 bits wide. This means the maximum address that the bus can hold is FFFF for a maximum address range of 64K bytes (65,536). Keep in mind, this uses the external address/data bus (A0:A15 for Address/Data and B10, 11, 12, 15) for Control.

BANK SWITCHING

If more than 64k of addressing is needed in the HPC system, the following method of increasing memory space can be used. Divide the total address range into two halves (32k bytes each). One half of this address range will be the MAIN memory address space. The MAIN memory address space will contain logical addresses (those addresses which the Program Counter can generate) in the range 8000 to FFFF and is accessed when A15 is a '1'. This includes the Interrupt vectors' and the Reset vector memory locations. The other half of the address range will be the BANK memory address space. The BANK memory address space will contain logical addresses in the range 0000 to 7FFF and is accessed when A15 is a '0'. This includes the on-chip I/O, registers, and RAM at locations 0000 to 01FF.

Now, four additional address lines are created using Port B pins (B8, B9, B13, B14). This prevents the use of the four timer synchronous outputs TS0–TS3 which are the alternate functions for these pins. The BANK memory is now addressed using A0:A14, B8, B9, B13, B14 and is accessed when A15 is a '0'. The BANK memory address space is now expanded to 512k bytes broken down into 16 individually selectable banks of 32k bytes each selected by these four bits of Port B.

A look at Table 1 and *Figure 1* quickly tells you that only one bank in the BANK memory space can share the logical address range 0000:7FFF at any one time. Therefore, programs running in the BANK memory address space can only directly access data and programs in the MAIN memory address space or in it's own bank (selected by B8, B9, B13, B14). On chip resources, which include RAM, I/O, and registers are mapped into logical addresses 0000 to 01FF. These logical addresses are in the BANK memory address space, but, since these addresses are considered to be always on-chip by the HPC, it never looks at the external address/data bus and will not read external memory in this range. Therefore, the first 256 bytes in each bank of memory in the BANK memory space will not be accessible by the HPC, but this address range (on chip resources) is directly accessible by any bank of memory in the BANK memory address space. This is why *Figure 1* shows a total available memory of 536.5k.

The interrupt vectors are mapped into logical addresses FFF0 to FFFF which are in the MAIN memory address space. Interrupts are handled properly if they occur while executing a program out of one of the banks of memory in BANK memory space, since the interrupt vector locations have A15 set to '1' which will allow access to the MAIN memory space. However, these interrupt vectors must either point to a routine in the MAIN memory address space which performs the interrupt service or point to code that selects the appropriate bank of memory in the BANK memory space and go there if the interrupt service routine is located there.

The stack must be located so that it can be directly accessible from anywhere in memory. It can be placed in the MAIN memory space or in the on-chip RAM. Programs and data storage that must be shared and directly accessed by all memory banks in the BANK memory space should also reside in the MAIN memory space.

HPC OPERATING MODES

The HPC must be configured to run in one of it's Expanded modes of operation by setting the EA bit in the PSW to be able to address the BANK memory range of 0000 to 7FFF. This memory expansion addressing scheme will work if the HPC is configured in either the Normal Expanded mode (EXM pin tied low) or ROMless Expanded mode (EXM pin tied high). The Normal mode differs from the ROMless mode only by the fact that the HPC will access the on-chip ROM for addresses in the range of E000 to FFFF (in the case of the HPC16083) and will access the external MAIN memory for addresses in the range of 8000 to DFFF.

The external data bus size is determined once, at reset, by sampling the state of $\overline{\text{HBE}}$ (B12). If $\overline{\text{HBE}}$ is high when sampled, the HPC enters 8-bit mode. In 8-bit mode, only pins A0–A7 are used to transfer data and pins A8–A15 continue to hold the most-significant eight bits of the address. So, only the lower eight bits of the address need to be latched externally (*Figure 2*). If HBE is low when sampled, the HPC enters 16-bit mode. In 16-bit mode, all 16 pins of Port A are used to transfer data as well as addresses. Two octal latches are then required externally to hold each address as it is issued by the HPC. The signal ALE from the HPC clocks the latches (*Figure 3*).

Keep in mind that if the external memory is configured as 8-bit memory, then the program stack must be in internal on-chip RAM because it has to be accessible as 16-bit words. If the external memory is configured as 16-bit memory then the stack can be in external RAM but must be in the MAIN memory address space to be directly accessible by all banks.

PROGRAMMING CONVENTIONS

A convention must be followed for maintaining linkages between the programs and data running in the MAIN memory space and the programs and data running in the BANK memory space. For the following discussion, the MAIN memory space will be referred to as just another bank of memory.

MAIN bank reserved portion

A portion of the MAIN memory bank should be reserved for Jump instructions to subroutines in the MAIN memory bank that need to be called by programs running in any selected bank in the BANK memory space. These Jump instructions serve as entry points for programs and subroutines. Typically, common functions that are required by programs running in several banks would be put in the MAIN memory bank. These could include: interrupt service routines, I/O drivers, and data handling and conversion routines. This portion also contains address pointers to tables of data in the MAIN memory bank that also are required by programs running in any selected bank in the BANK memory space. See Listing 1 for an example.

BANK memory reserved portion

A portion of each bank in the BANK memory space should be reserved for Jump instructions to subroutines in that bank that need to be called by programs running in the MAIN memory bank. These Jump instructions serve as entry points for programs and subroutines. For example, each bank in the BANK memory space could contain routines that perform unique but related functions. One bank could be reserved for math routines; another bank could perform message handling; and yet another could contain diagnostic routines. All of these functions could be scheduled and executed from some sort of Supervisor running in the MAIN memory bank performing the linkages to all these routines thru the entry points. This reserved portion of each bank also contains address pointers to tables of data in that bank that also are required by programs running in the MAIN memory bank. In the case of a bank running message handling routines, address pointers could be inserted to point to buffers that programs running in MAIN memory need to access. See Listing 2 for an example.

Linkage areas

These reserved portions of each memory bank (MAIN space or BANK space) must be fixed and known to each other memory bank that requires access to programs and data in that bank. Therefore, one other requirement in each bank is a set of labels that are assigned the values of the pointer locations to subroutines and tables in the bank of interest (see Listings 3 and 4).

One last requirement in the MAIN memory bank, if it is to perform bank to bank moves and for general housekeeping, is to reserve two byte locations to be used to keep track of the bank currently selected (high byte value on Port B) being used in the transfer of data (see Listing 5).

From the MAIN memory bank, the user can access all memory in the system. He can call subroutines in any bank in the BANK memory space and read/write data to the entire memory. From any bank in the BANK memory space, the user can call subroutines in the MAIN memory bank and read/write data to the MAIN memory bank in addition to his own local bank.

The basic procedure used to call a program in the BANK memory space from the MAIN memory bank is merely to set the proper value on the Port B select lines and execute a Jump to SubRoutine through a pointer in the selected bank:

Interrupts

Regardless of where the interrupt service routine actually resides, an image of the bank selected must be retained by the service routine to allow it to return to the appropriate bank when complete. If the interrupt service routine is in the MAIN memory bank, the linkage is handled in the normal fashion where the interrupt vector points to the service routine. The interrupt service can reside in the BANK memory space and takes a little extra overhead for the linkage.

To call a program in the MAIN memory bank from the BANK memory space, merely execute a Jump to SubRoutine through a pointer in the MAIN memory bank:

```
JSRL CMPBLNK        ;see Listing 1 and 4
```

EXAMPLE SOFTWARE

Now that a convention has been established for communicating between the MAIN memory space and the BANK memory space, let's take a look at some sample code that can be used to move data between these memory spaces. In order to make the selection of bank memory efficient, it is important to keep in mind that the four bits of the high byte of Port B that are used to select a bank of memory in the BANK memory space can be written to directly since the other 4 bits of this byte of Port B are used for memory control outputs (the external control bus) and are not affected by a write to the high byte of Port B.

Bank to Bank data transfer by MAIN

Listing 6 shows the setup required to initialize the linkage area in order to perform a transfer of data from one bank to another bank in the BANK memory space by a program running in the MAIN memory space. This involves setting up the RAM locations that are used to 'select' the source bank and the destination bank, select the source bank to determine the starting address of the area to move, select the destination bank to determine the starting address of the area to move data into, then finally calling the subroutine in MAIN memory that performs the move. After the setup portion, the subroutine that performs the transfer is presented. This code assumes that the external memory is configured in 16-bit mode.

Bank to MAIN data transfer by Bank

Listing 7 presents a similar example for moving blocks of data from a bank in BANK memory to MAIN memory by a program running in that bank. This code also assumes that the external memory is configured in 16-bit mode.

External 8-bit mode

If the external memory is configured in 8-bit mode, the setup portion changes because the initialization of the RAM address pointers SSTART, DSTART and DEND requires building word address pointers from word pointers in the external reserved areas of each bank. In 8-bit mode, this requires two 8-bit transfers compared to one 16-bit transfer in 16-bit mode (see Listing 8). Once these address pointers have been built, however, the subroutine that actually performs the move does not have to change because 1) word transfers are allowed between On-chip RAM and registers regardless of the mode and 2) the subroutine performs byte moves. To improve speed in the 16-bit mode, this subroutine can be modified to perform 16-bit moves. However, keep in mind that this will impose the restriction on the address pointers in the linkage areas of requiring that addresses be on word boundaries. Listing 9 presents a similar example for moving blocks of data from a bank in BANK memory to MAIN memory by a program running in that bank.

PROGRAM DEVELOPMENT

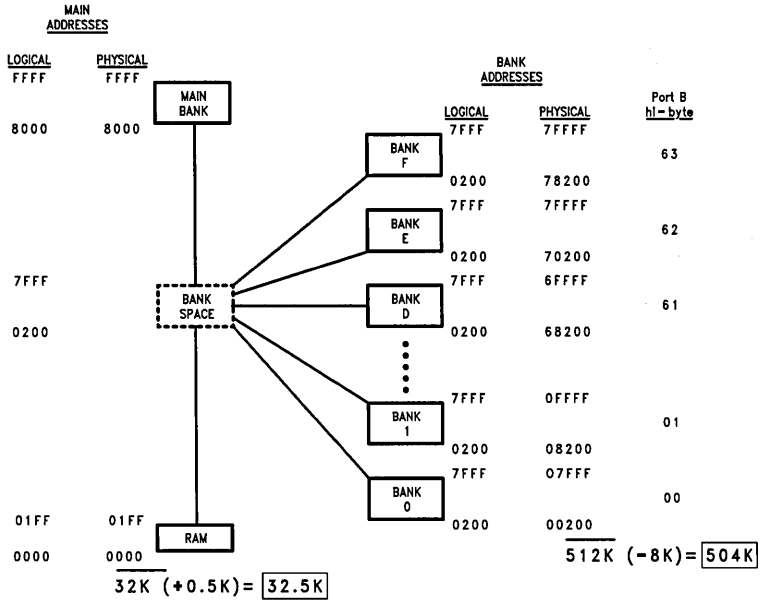
The MOLE monitor software can support the development of HPC programs in multiple banks of memory. It provides the means of qualifying a trigger condition, as set in Trace or Breakpoint functions, with the memory bank number. The BANK command will allow a trigger only when executing in the memory bank of interest. The MOLE supports a total of 16 memory banks which are normally selected by 4 bits of Port B as described earlier. See the HPC Personality Board User's Manual for further detail on this command.

CONCLUSION

What has been presented is a method to expand the memory space of the HPC to 544k. Although this method utilized four bits of Port B to accomplish the extra addressing, theoretically, the remaining 8 bits could have been used if not required for other purposes. This could mean a maximum addressability for the HPC of greater than 128 Megabytes. However, the MOLE will only support the fixed definition of four extra address lines. Clever utilization of existing resources can enable you to get the most out of hardware and software limited only by one's imagination.

TABLE I. Logical Addresses vs Physical Memory Locations

Logical Address	Bank #	Hi Byte Port B	Physical Address	
0000:7FFF	0	00	00000:07FFF	
0000:7FFF	1	01	08000:0FFFF	
0000:7FFF	2	02	10000:17FFF	
0000:7FFF	3	03	18000:1FFFF	
0000:7FFF	4	20	20000:27FFF	
0000:7FFF	5	21	28000:2FFFF	
0000:7FFF	6	22	30000:37FFF	
0000:7FFF	7	23	38000:3FFFF	(BANK)
0000:7FFF	8	40	40000:47FFF	
0000:7FFF	9	41	48000:4FFFF	
0000:7FFF	A	42	50000:57FFF	
0000:7FFF	B	43	58000:5FFFF	
0000:7FFF	C	60	60000:67FFF	
0000:7FFF	D	61	68000:6FFFF	
0000:7FFF	E	62	70000:77FFF	
0000:7FFF	F	63	78000:7FFFF	
8000:FFFF	—	—	08000:0FFFF	(MAIN)



TL/DD/9342-1

FIGURE 1. How BANK Memory is Mapped into the HPC Address Space

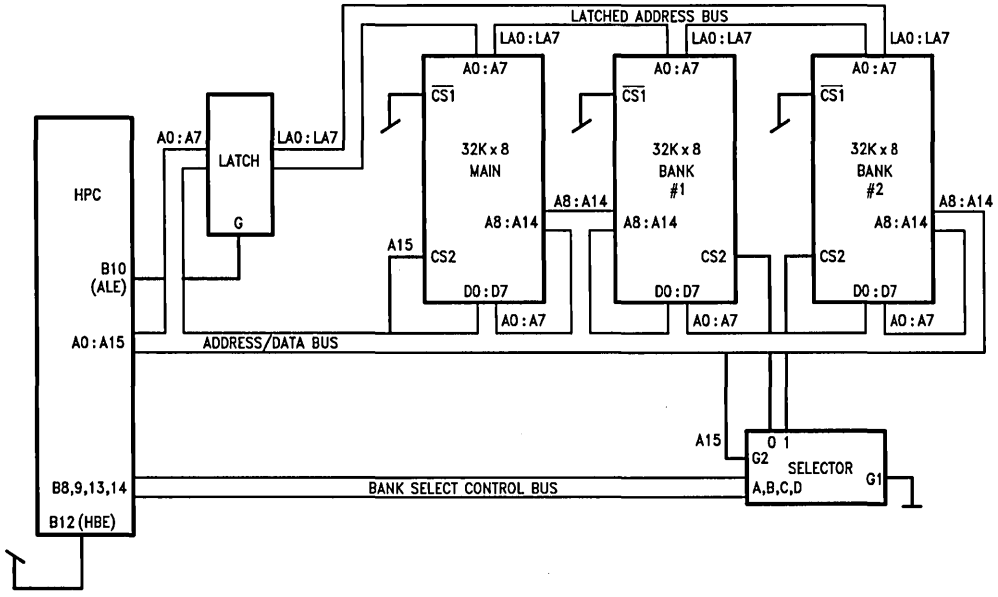
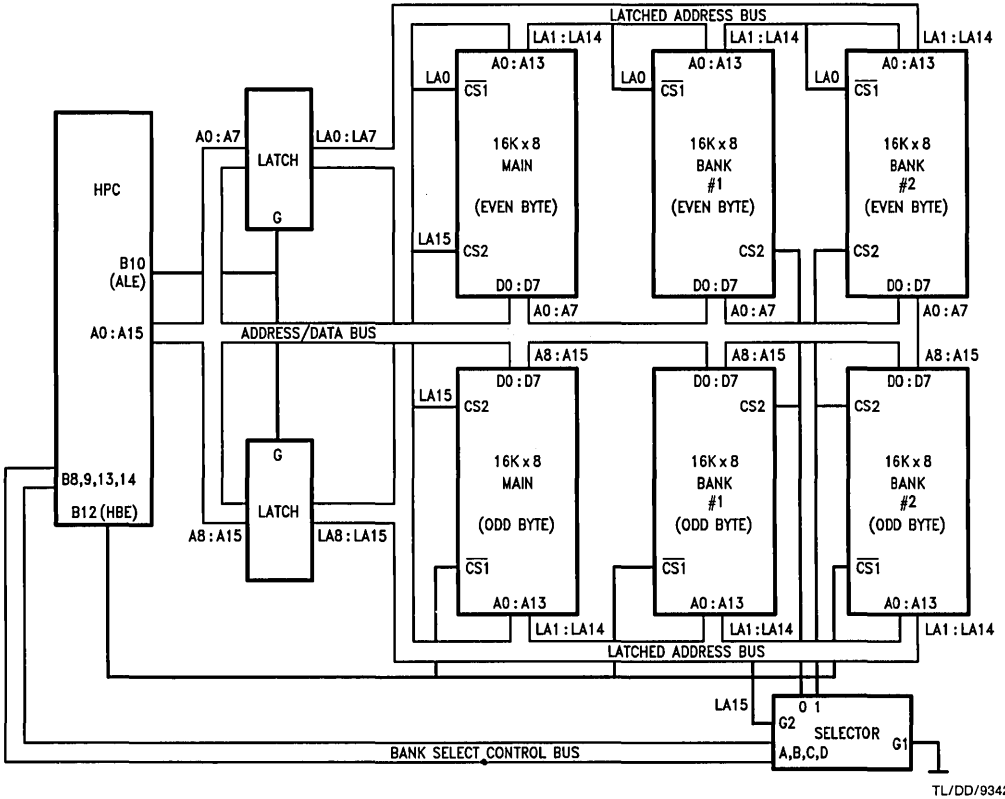


FIGURE 2. HPC in 8-Bit Mode

TL/DD/9342-2



TL/DD/9342-3

FIGURE 3. HPC in 16-Bit Mode

5

```
            .=08000      ;set PC counter to 8000
;This code resides in the MAIN memory bank
;
;     The following address pointers are inserted to allow
;     programs running in BANK memory to find these
;     locations. They represent the starting and ending
;     location for code in MAIN memory.
;
      .WORD INIT   ;addr pointer to first location in bank
      .WORD PROGEND      ;addr pointer to last location in bank
;
;     The following Jump instructions are inserted to allow
;     programs running in BANK memory to call these
;     subroutines. They represent subroutines that compare
;     blocks of memory in MAIN memory space with blocks of
;     memory in BANK memory space or compare blocks of memory
;     in BANK memory for zeros.
;
      JMPL CMPM          ;entry for compare blocks (MAIN-BANK)
      JMPL CMPBFB        ;entry for compare BANK cleared
```

LISTING 1. MAIN Bank Reserved Portion

```
            .=0200       ;set PC counter to 200
;This code resides in any bank in BANK memory
;
;     The following address pointers are inserted to allow
;     programs running in MAIN memory to find these
;     locations. They represent the ending location for code
;     in this bank of BANK memory.
;
      .WORD PROGEND      ;addr pointer to last loc in this bank
;
;     The following Jump instructions are inserted to allow
;     programs running in MAIN memory to call these
;     subroutines. They represent subroutines that compare
;     blocks of memory in MAIN memory space with blocks of
;     memory in this bank, diagnostic routines, and interrupt service routine.
;
      JMPL CMPMB   ;entry for comp blocks (MAIN-this bank)
      JMPL BTEST   ;entry for this bank's diag routines
      JMPL BINTS   ;entry for this bank's interrupt service routine
```

LISTING 2. Typical Bank Reserved Portion

```
;This code resides in the MAIN memory bank
;
;     linkages to Bank 0
;
B0START = 0200          ;addr of pointer to first avail loc
;
CMPMB0 = 0202           ;addr of JMPL to routine that compares
;                            move results
B0TEST = 0205           ;addr of JMPL to test routines
;
;     linkages to Bank 1
;
B1START = 0200          ;addr of pointer to first avail loc
;
CMPMB1 = 0202           ;addr of JMPL to routine that compares
;                            move results
B1TEST = 0205           ;addr of JMPL to test routines
;
;     linkages to Bank 2
;
B2START = 0200          ;addr of pointer to first avail loc
;
CMPMB2 = 0202           ;addr of JMPL to routine that compares
;                            move results
       = 0205           ;addr of JMPL to test routines
;
B2INTS = 0208           ;addr of JMPL to interrupt service routine
```

LISTING 3. MAIN Memory Bank Linkage Area

```
;This code resides in any bank in BANK memory
;
;     linkages to MAIN memory
;
MSTART = 08000          ;addr of pointer to first avail loc
MEND = 08002            ;addr of pointer to last avail loc
;
CMPM = 08004            ;addr of JMPL to routine that compares
;                            move results
CMPBLNK = 08007         ;addr of JMPL to routine that compares
;                            if a block in selected BANK is zero
```

LISTING 4. Typical Bank Linkage Area

```
;This code resides in the MAIN memory bank
;
; The following locations are used for bank to bank moves
; and compares
      BANKS = 01C0          ;Source bank byte value
      BANKD = 01C1          ;destination bank byte value
;
      BANK0 = 0         ;Port B high byte value to select bank 0
      BANK1 = 1         ;                                      1
      BANK2 = 2         ;                                      2
      BANK3 = 3         ;                                      3
      BANK4 = 020       ;                                      4
      BANK5 = 021       ;                                      5
      BANK6 = 022       ;                                      6
      BANK7 = 023       ;                                      7
      BANK8 = 040       ;                                      8
      BANK9 = 041       ;                                      9
      BANKA = 042       ;                                      10
      BANKB = 043       ;                                      11
      BANKC = 060       ;                                      12
      BANKD = 061       ;                                      13
      BANKE = 062       ;                                      14
      BANKF = 063       ;                                      15
;
;     Main Memory Bank is logical and physical address range
;     8000:FFFF. Switched Memory Banks are logical addresses
;     in the range 0000:7FFF combined with the
;     Port B(14,13,9,8) bits to create physical addresses in
;     the range 00000:7FFFF
;
```

LISTING 5. BANK Memory Management

```
        LD M(0E3),BANK1 ;set bank select lines to select bank 1
        JSRL B1TEST     ;see Listing 2 and 3
                                        •
                                        •
                                        •
INT35:
LD      BANKS,M(0E3)    ;save bank interrupted from
LD      M(0E3),BANK2    ;set bank select lines to select bank 2
JSRL    B2INTS          ;see listing 2 and 3
                                        •
                                        •
                                        •
LD      M(0E3),BANKS    ;restore bank interrupted from
RETI
                                        •
                                        •
                                        •
.IPT    2,INT35         ;set interrupt vector
```

```
;This code resides in the MAIN memory bank
;
      LD M(BANKS),BANK0       ;prepare to move data from Bank 0
      LD M(BANKD),BANK1       ;to Bank 1
      LD M(0E3),BANK0          ;select Bank 0
      LD W(SSTART),W(B0START) ;set starting address in source bank
      LD M(0E3),BANK1          ;select Bank 1
      LD W(DSTART),W(B1START) ;set starting address in destination bank
      LD W(DEND),W(B1START)   ;set ending address in destination bank
      ADD W(DEND),1023         ;to 1K greater than starting address
      JSRL MOVBB               ;do it
            •
            •
            •
            •
            •
;
;  This subroutine moves data from bank memory to bank memory
;  where the source bank is defined by the contents of the byte
;  at RAM location BANKS and the destination bank is defined by
;  the contents of the byte at RAM location BANKD. In addition,
;  the following locations must be set up before calling:
;
;  SSTART  →  RAM location containing source bank start address
;  DSTART  →  RAM location containing destination bank start address
;  DEND  →  RAM location containing destination bank end address
;
MOVBB:
      LD B,W(DSTART)          ;B ← starting address (destination)
      LD K,W(DEND)            ;K ← ending address (destination)
      LD X,W(SSTART)          ;X ← starting address (source)
LOOPBB:
      LD M(0E3),M(BANKS)      ;select source BANK
      LD A,M(X+)               ;byte at source into A
                               ;increment source pointer
      LD M(0E3),M(BANKD)      ;select destination BANK
      XS A,M(B+)               ;A into byte at destination, bump pntr
      JP LOOPBB                ;back for more if B less than K
      RET
```

LISTING 6. Move Data by MAIN from BANK to BANK (16-Bit Mode)

```
;This code resides in any bank in BANK memory
;
      LD W(SSTART),TABLE1     ;starting address of table in this memory
      LD W(DSTART),W(MSTART)  ;starting address in main memory
      LD W(DDEND),TABLE1+1023 ;ending address in main memory
      JSRL MOVE               ;do it
            •
            •
            •
            •
            •
;
;  This subroutine moves data from this bank to main memory
;
;  SSTART  →  RAM location containing source memory start address
;  DSTART  →  RAM location containing destination memory start addr
;  DEND →  RAM location containing destination memory end address
;
MOVE:
      LD B,W(DSTART)          ;B ← starting address (destination)
      LD K,W(DEND)            ;K ← ending address (destination)
      LD X,W(SSTART)          ;X ← starting address (source)
LOOPBM:
      LD A,M(X+)              ;byte at source into A
                               ;increment source pointer
      XS A,M(B+)              ;A into byte at destination, bump pntr
      JP LOOPBM               ;back for more if B less than K
      RET
```

LISTING 7. Move Data by BANK from BANK to MAIN (16-Bit Mode)

```
;This code resides in the MAIN memory bank
;
      LD M(BANKS),BANK0       ;prepare to move data from Bank 0
      LD M(BANKD),BANK1       ;to Bank 1
      LD M(0E3),BANK0          ;select Bank 0
      LD M(SSTART),M(B0START) ;set starting address in source bank
      LD M(SSTART+1),M(B0START+1)
      LD M(0E3),BANK1          ;select Bank 1
      LD M(DSTART),M(B1START)  ;set starting address in destination bank
      LD M(DSTART+1),M(B1START+1)
      LD M(DEND),M(B1START)   ;set ending address in destination bank
      LD M(DEND+1),M(B1START+1)
      ADD M(DEND),L(1023)     ;to 1K greater than starting address
      ADC M(DEND+1),H(1023)
      JSRL MOVBB               ;do it
                  •
                  •
                  •
                  •
                  •
;
;  This subroutine moves data from bank memory to bank memory
;  where the source bank is defined by the contents of the byte
;  at RAM location BANKS and the destination bank is defined by
;  the contents of the byte at RAM location BANKD. In addition,
;  the following locations must be set up before calling:
;
;  SSTART --> RAM location containing source bank start address
;  DSTART --> RAM location containing destination bank start address
;  DEND --> RAM location containing destination bank end address
;
MOVBB:
      LD B,W(DSTART)          ;B <-- starting address (destination)
      LD K,W(DEND)            ;K <-- ending address (destination)
      LD X,W(SSTART)          ;X <-- starting address (source)
LOOPBB:
      LD M(0E3),M(BANKS)      ;select source BANK
      LD A,M(X+)                ;byte at source into A
                                 ;increment source pointer
      LD M(0E3),M(BANKD)      ;select destination BANK
      XS A,M(B+)                ;A into byte at destination, bump pntr
      JP LOOPBB                 ;back for more if B less than K
      RET
```

LISTING 8. Move Data by MAIN from BANK to BANK (8-Bit Mode)

```
;This code resides in any bank in BANK memory
;
      LD M(SSTART),L(TABLE1)  ;starting address of table in this memory
      LD M(SSTART+1),H(TABLE1)
      LD M(DSTART),M(MSTART)  ;starting address in main memory
      LD M(DSTART+1),M(MSTART+1)
      LD M(DEND),M(MSTART)    ;set ending address in main memory
      LD M(DEND+1),M(MSTART+1)
      ADD M(DEND),L(1023)      ;to 1K greater than starting address
      ADC M(DEND+1),H(1023)
      JSRL MOVE                  ;do it
            •
            •
            •
            •
            •
;
;  This subroutine moves data from this bank to main memory
;
;  SSTART  →  RAM location containing source memory start address
;  DSTART  →  RAM location containing destination memory start addr
;  DDEND   →  RAM location containing destination memory end address
;
MOVE:
      LD B,W(DSTART)          ;B ← starting address (destination)
      LD K,W(DEND)            ;K ← ending address (destination)
      LD X,W(SSTART)          ;X ← starting address (source)
LOOPBM:
      LD A,M(X+)              ;byte at source into A
                               ;increment source pointer
      XS A,M(B+)              ;A into byte at destination, bump pntr
      JP LOOPBM               ;back for more if B less than K
      RET
```

LISTING 9. Move Data by BANK from BANK to MAIN (8-Bit Mode)

Assembly Language Programming for the HPC™

National Semiconductor
Application Note 510
Steve McRobert

HOW TO WRITE SHORT, EFFICIENT, BUT UNDERSTANDABLE ASSEMBLER PROGRAMS

INTRODUCTION

One of the design objectives of the HPC family was that it should be very easy to use. With this in mind the instruction set has been designed so that it obeys a very simple set of rules. Once these rules have been learned, the programmer can write code with very little reference to instruction manuals.

The HPC is fully memory mapped. Every piece of hardware attached to an HPC core appears as a byte or a word in a linear 64K byte address space. Any data movement or arithmetic instruction can operate on any memory location and everything in the HPC has a memory location, including the accumulator. All of the I/O ports, the peripheral control registers, RAM and ROM are treated in exactly the same fashion as far as the assembly language programmer is concerned.

The HPC assembly language syntax can be explained by describing the instruction codes and the addressing modes. The instruction code tells the processor what operation it is performing, such as an add, a subtract, a multiply, a divide or a data movement instruction. The addressing mode is the way that the programmer specifies the value or values to be operated on to the microprocessor itself.

ADDRESSING MODES

Operations can be performed on any memory location. One can, for example, increment or decrement any byte or word of any memory location in the HPC. Increment and decrement are examples of single address instructions. These are instructions which have only one operand. Other examples are the bit set, bit test and bit clear instructions. These five instructions are good examples of the basic thinking behind the HPC instruction set. All of these instructions use the same four addressing modes.

Direct

The simplest addressing mode to understand is that known as direct. In this mode the address of the variable to be operated on is included as part of the sequence of bytes that comprises the entire instruction. For example, in order to perform a decrement on memory location 0F0 this value is included in the string of bytes that forms the instruction.

```
Examples:
 DECSZ   0F0.B
 INC     0F0.W
```

The increment instruction, like most other instructions with HPC, can operate on either a byte or a word. A byte access is specified by putting a B after the address of the variable, a word access by writing W.

Register Indirect

This addressing mode usually generates less bytes of code than any other. HPC has two 16-bit registers, B and X, which can be used as general purpose memory locations but also have a specific function as pointers to memory. These instructions take up very little ROM space because the address of the variable to be operated on is contained in the pointer register and the pointer register to be used is specified as part of the instruction. An instruction such as increment, using register indirect, can thus be only 1 byte long as it does not need to be followed by a byte specifying the address of the variable.

```
Examples:
 INC     [B].B ;byte increment, B pointer
 INC     [X].W ;word increment, X pointer
```

Indirect

B and X provide two 16-bit pointers to memory. Programmers will often wish to have more than two pointers in use at any one time. HPC therefore provides indirect addressing mode. In this mode a 16-bit pointer to the location to be accessed is stored in the basepage of the HPC. The instruction, therefore, is followed by a single byte which specifies the address of this 16-bit pointer. The bottom 192 bytes of RAM are on chip with the HPC and are in the so-called base page. The base page is normally used for storing frequently accessed variables as only a single byte of address is required to access a base page variable. When using indirect addressing mode, the 16-bit pointer value must always be in the base page.

```
Examples:
 DECSZ   [0].W    ;decrement a word
 INC     [0FE].B ;increment a byte
```

The base page is in the region of 0 to 0FF bytes. This area also contains the most frequently used registers such as the accumulator. The programmer can thus use indirect addressing mode with registers such as the accumulator acting as the pointer. This is an example of the simplicity of the HPC instruction set. Any operation can be performed on any HPC register simply by invoking its address in the HPC 64 kbyte addressing space.

Indexed

The last of the four basic addressing modes is indexed mode. Indexed is very similar to indirect except that an 8- or 16-bit immediate value follows the address of the 16-bit pointer and is added to it to generate the address of the variable to be accessed. This allows a table of values to be located anywhere in memory and the pointer register need only be implemented or decremented to move through the table of values.

```
Examples:
 INC     0FF00 [4].W   ;increment a word
 DECSZ   02 [2].B      ;decrement a byte
```

Bit Operations

The bit operations of the HPC allow any bit in the memory of the HPC to be accessed. The addressing modes for these three operations, SBIT, RBIT and IFBIT, always refer to the memory location as a byte. The individual bit of the byte to be tested, using the four addressing modes already described, is actually coded into the opcode itself. This could be described as an implied addressing mode but this definition is not normally used in HPC. The way this works can be seen from the opcode map in the programmers guide of the HPC, where it can be seen that there are in fact eight opcodes shown for each of the three different bit instructions.

```
Example:
 SBIT 5, 2.B   ;set bit 5 of byte
               ;at address 2.
```

Double Register Indirect

A rule of thumb when trying to decide which addressing mode one can use with which opcode in HPC is that you can use any combination of addressing mode and opcode that is sensible. An example of this is a special addressing mode which works only for the bit instructions. This addressing mode is known as double register indirect and uses a combination of the B and X registers to index into any bit of a 64k bit string, the lower boundary of which can be located anywhere in memory.

When using this addressing mode the B register points to the lowest byte of this 8k byte string, while the most significant 13 bits of the X register point at the individual byte in the string that is being accessed. The three least significant bits of the X register point at the bit of the byte that the instruction is pointing at. By using this addressing mode, words of any length can be scanned for whether individual bits are set or cleared. This addressing mode, while unusual, fits into the scheme of things as it clearly is only of any relevance to the individual bit instructions.

```
Examples:
 SBIT X,    [B].B; Set bit
 IFBIT X,   [B] B; test bit
```

Note that the bit instructions only operate on bytes, to allow operations on words would require twice as many opcodes for no gain.

Two Address Instructions

The five instructions described so far have only one operand. There are many more instructions in the HPC instruction set which have two operands, such as arithmetic instructions, the comparison instructions and data movement instructions. The HPC instruction set allows any of these instructions to use any of the four addressing modes already described. An instruction such as multiply, for example, when written in the HPC assembler syntax as shown below shows the opcode followed by the destination operand, which is then followed by the source operand. The result of the operation in all cases except the comparison instructions winds up in the destination operand. The comparison instructions, IFEQ and IFGT do not affect the values of any memory location but, like all other two operand instructions, can operate on any two words or bytes in the HPC addressing space.

```
Examples:
 MUL A, [B].B
 MUL .0.W,2.W
```

The destination operand in HPC may be either the accumulator or a byte or word of memory accessed using the direct addressing mode. If the destination operand is the accumulator, the source operand may be addressed using direct, register indirect, indirect or indexed addressing modes as well as the familiar immediate addressing mode. The programmer can thus load the accumulator with an 8- or 16-bit immediate value which follows the opcode, multiply the accumulator with that value, divide the accumulator by that value or compare the accumulator by that value. Using the accumulator as the destination operand gives maximum flexibility in the choice of addressing mode for the source operand and also tends to produce a shorter instruction in terms of its length in bytes as the opcode does not have to include the address of the destination operand.

```
Examples:
 LD A, #37              ;load A With
                        ;immediate values.
 add 0FE.W,#  0F000     ;Add immediate to
                        ;memory.
```

Instruction Lengths

Tables are provided in the HPC users manual to allow the user to estimate the number of bytes an instruction will use and the time this instruction will take to execute. To use these tables the programmer must be aware of the name of the addressing mode he is using. This is perfectly clear for the single address instructions described at the beginning of this note but perhaps needs some explanation for two operand instructions.

For two operand instructions with the accumulator as the destination, the addressing mode is named after that used for the source operand. For example, load accumulator using a value pointed at by indirect addressing mode is referred to simply as indirect addressing mode.

Operations on Direct Memory

There are two addressing modes which allow operations to be performed directly on memory locations. If the destination operand is directly addressed memory, then the source operand may be directly addressed memory or an immediate value. These two are the only combinations of addressing modes that can be used where the destination operand is a memory location.

```
Examples:
 DIV   010.W, 0F000.W
 direct-direct mode

 DIV   0F0.B,#10
 immediate direct mode.
```

Special Symbols

Some special symbols have been allocated in the HPC cross assembler. These are A, B, K, X, PC and SP. The programmer can also define his own symbols using the equals directive of the assembler. The way that the symbols described above would be defined using the equals directive are shown below by way of example.

```
Example:
  A = 0C8.W
  B = 0CC.W
  X = 0CE.W
  K = 0CA.W
 PC = 0C6.W
 SP = 0C4.W
```

Note that these symbols cannot be redefined so the above set of definitions should never be included in a user program.

IMPLIED ADDRESSING MODES

Some of the HPC's opcodes have been shortened by using implied addressing mode. A few examples have already been shown. This section describes some more special cases. It could be said that accumulator as destination is an example of an implied addressing mode, where the address of the destination is coded into the instruction. There are some special purpose instructions which use implied addressing mode for instructions which are used very frequently. In most cases these instructions look exactly the same to the programmer as instructions using the addressing modes described earlier. For example there is a special opcode for load B with an immediate value. The programmer could do this using the immediate direct addressing mode but a special opcode has been provided to make this instruction shorter.

Load B and K is a special immediate load which loads both the B and K registers in one operation.

Carry Flag

The carry flag may be accessed using the standard bit test instructions because it can be read in the processor status word, but as carry must so often be set and tested, special instructions to do this have been included which do not require the address of the carry flag.

Multiply and Divide

Finally, the divide double and multiply instructions both have to manipulate 32-bit values. These therefore have to store an operand in two concatenated registers. The HPC instruction set cannot specify two registers with one address. Therefore these instructions default to using the X register as the high word of their 32-bit value.

The source and destination of a multiply instruction are specified as normal except that the 32-bit answer is stored in the destination operand with the 16 high bits of the answer stored in the X register. The divide double instruction basically performs the inverse of multiply, taking the 32-bit value formed by X concatenated with the destination value and dividing it by the source value. Divide double, like divide, yields a 16-bit result and a 16-bit remainder. For both divide double and divide the remainder is stored in the X register. In both cases the K register is used for intermediate value storage and is cleared as a result of this operation.

As the result of divide double can only be a 16-bit value, a full 32-bit divide is performed by following a 16-bit divide with a 32-bit divide as shown below. The example below shows how the divide instructions work together and also highlights the combinations of addressing modes that can and cannot be used with HPC.

```
                    LD          B,#11
                    DIV         HIGH.W,#
10
LOOP:               DIVD        LOW.W,#1
0
                    LD A, X
                    ST A, [B]
                    DECSZ B
                    JP LOOP
```

This example shows the conversion of a 32-bit binary value in words low and high into a 10-digit BCD number in the 10 bytes starting from 1. The conversion is performed one digit at a time and the B register is used to point at the byte's location where the digit is to be stored. The first instruction of the programme therefore is to initialize the B register. The divide instruction divides word high by 10 using immediate direct addressing mode and stores the answer back in word high. The remainder is stored in the X register. The divide double instruction then divides X concatenated with word low by 10. Because X contains a remainder, the result of this division will always be a 16-bit value and can thus be stored in word low. The remainder is stored in X and is in fact the modulus and is thus the BCD digit that we have derived on this pass through the numbers.

We now wish to store the remainder into one of our BCD digit locations using register indirect mode. We need to load the value into the accumulator from X. The X register is nothing special in this application, so load A with word X is in fact an example of direct addressing mode.

Now that our BCD value is in the accumulator, we can store this in the byte location using B register indirect addressing mode.

The next instruction is decrement skip on zero. This uses direct addressing mode to decrement the B register. This instruction is an example of many in HPC which perform more than one function. As well as decrementing the memory location specified, this instruction also compares it with zero after the decrement has been performed. If the result is zero, the instruction following the decrement skip on zero instruction is skipped. That is to say it is ignored and control passes to the instruction following it. In this example the final instruction of the routine is a single byte jump back to the divide instruction. The overall loop is executed ten times in order to perform the conversion. On the final pass through the loop, B becomes zero and execution of this algorithm is terminated.

Auto Increment/Decrement Instructions

This multi-function instruction capability is best illustrated by the four special addressing modes register increment or decrement with or without conditional skip, which work only with the data movement instructions load and exchange. The load instruction in general uses any of the five two-address modes or the two combination modes to transfer data from one location to another.

The exchange instruction is similar except that the destination must always be the accumulator. Exchange not only takes the source and puts the value into the destination but also takes the value from destination and puts it into source. Clearly there is no immediate addressing mode for exchange as a destination cannot be stored into an immediate value.

When load and exchange are used with the X register as a pointer and register indirect mode, a suffix + or − can be added after the X. In this case, once the data movement operation has been performed, the X register is incremented or decremented by one or two according to whether

there has been a byte or a word access respectively. A further refinement on this is provided by the load and exchange with conditional skip instructions, LDS and XS respectively. These only work with the B register as the pointer and perform two more operations rather similar to the decrement skip on zero instruction. Once the increment or decrement has been performed, the B register is compared with the K register, otherwise known as the limit register. If an increment has been performed and B is greater than K, the instruction following the movement instruction will be skipped. If a decrement is performed, the instruction is skipped if B is less than K.

An example of how these specialized instructions are used is given by the block move routine shown below;

```
        LD   X,#START
        LD   BK,#BEGIN,#END
LOOP:   LD   A, [X+].W
        XS   A, [B+].W
        JS   LOOP
```

This routine moves a block of data from one location to another. The X register is initialized first and is used as a pointer to the first value to be moved in the source block. The B and K registers point to the first and last values respectively in the destination block. The loop itself consists of only three bytes. The first instruction loads the accumulator with the word pointed to by the X register and increments X by two. A second instruction exchanges the accumulator with the word pointed to by the B register, increments the B register by two and compares it with K. If B is greater than K, the jump instruction is skipped and this loop is terminated.

The example shows how HPC code can perform a great deal with very few instructions and use up very few bytes of code while doing so.

These auto increment/decrement instructions are the only examples where an addressing mode cannot be used for any instruction where it might make sense. It is however fairly easy to remember which addressing modes these can be used with. Auto increment/decrement can be used with the load and exchange instructions for the X register. Auto increment or decrement with conditional skip can be used with load and exchange instructions using the B register as a pointer. No other combinations are allowed.

We have not provided specific string move or search instructions but the auto increment/decrement operations provide building blocks allowing the programmer to assemble his own stock. In the block move instruction shown above, the value being moved is in the accumulator in between the load and exchange instructions. The programmer can then compare this value with anything he wishes, fill BCD to ASCII, pack BCD, unpack BCD or perform any operation he likes on a string of data.

HPC ASSEMBLY CODE

The addressing modes usable for each opcode are described in a shorthand form.

```
Example:
            ADD  MA < MA + MemI
```

In the above syntax MA means directly addressed memory or the accumulator and MemI means memory addressed using any of the four basic single-address addressing modes or an immediate value. This would be better written as shown below:

```
            A < A + MemI
or          M < M + M
or          M < M + I
```

Expanding the syntax highlights that the flexible addressing modes such as register indirect may only be used if the destination is the accumulator. It also shows that if the destination is direct memory the source may only be an immediate value or another direct memory location.

When writing assembly code the programmer writes the same mnemonic whether a memory location is a piece of RAM or ROM or an I/O port or the accumulator. In general any source or destination variable may be a byte or a word and combinations are allowed. Care must be taken when storing word into a byte location that the programmer really wishes to truncate that value to byte and throw away the upper 8 bits of the value. When loading a byte into a word location the upper 8 bits of the word location will be filled with zeros. If memory external to the HPC is used, this may be 8 or 16 bits wide. The programmer must be aware of this when writing his assembly language as HPC cannot cope with the programmer requesting a 16-bit access to 8-bit wide external memory. The HPC will not convert this to two sequential 8-bit accesses.

The only exception to this rule is that a pointer word in indirect or indexed addressing modes must always be in the base page. This is because only one byte has been allowed in the overall length of the instruction for the address of the pointer.

For all other addressing modes there is no difference in the assembly language the programmer writes between accessing a variable that is in the base page and a variable that is above address 0FF.

The programmer should be aware however that variables in the base page consume less bytes per access and the instruction will execute more quickly than non-base page variables. When studying the data sheet to see how long an instruction is, the programmer will see that the table result is different according to whether variables are base page or not. The programmer should therefore allocate base page to variables which are used most often.

EXECUTION SPEED

There are 64 bytes of RAM above the base page. These, like the base page RAM, require zero wait states to access even when the processor is running at full speed. They do however require 2 bytes of code for their addresses. These

64 bytes may best be made use of by using them as the stack area as the 16-bit stack pointer contains the full address and therefore there is no penalty in instruction length in putting the stack in this non-base page on-chip RAM.

Note that there is no difference in execution time between byte and word accesses, that is to say accesses to byte or word variables. When studying the data sheet, differences in program length and therefore in execution time will be observed according to whether the address of a directly addressed variable is a byte or a word. It is important to understand the difference between the width of the variable and the width of the address that is used to access that variable.

The cycles per instruction table is not always clear about the number of wait states applied to different variables. The HPC includes a wait state register which sets the number of wait states to be used when accessing external memory, the internal ROM, or internal registers associated with ports A and B. Wait states may be applied to these on-chip registers to allow compatibility with development tools such as the MOLE™ and HPC Designer Kit board, as when these tools are run on high clock speeds wait states must be applied for accesses to the port recreation logic. The HPC needs wait states for accessing slow external memory and when running at high clock rates.

These wait states may be applied in order that the MOLE can provide a perfect emulation of a single-chip HPC. In the MOLE the HPC is running with external memory and thus the A port and some of the B port are used for address/data and control lines respectively. The A port and part of the B port must therefore be recreated external to the HPC. In the case of the MOLE this is done using a large array of PAL®s. Because they are external to the HPC, one wait state must be applied when accessing these externally recreated ports at high clock speeds. If wait states could not be applied to these ports in a masked ROM HPC, the MOLE would not be able to provide full speed emulation. This is just one example of how the design of the HPC has been influenced by the need to emulate it 100% exactly at full speed. Apart from this no wait states are applied to any access to address locations below 200 HEX, regardless of the addressing mode used.

The HPC data sheet does not make it clear how many wait states are applied when register indirect addressing mode is used. It implies that wait states are always applied when register indirect or similar addressing modes are used, but this is not the case.

The best way to time a piece of code is to write the code and then run it through the cross assembler to generate a source plus object listing. The number of bytes generated by each instruction can then be easily read and only the cycles and accesses table need be looked up in order to calculate how long each instruction takes to execute.

Note that accesses to internal ROM are subject to at least one wait state for exactly the same reason as accesses to the A or B ports.

SUMMARY

The HPC is fully memory mapped. The I/O Ports, Peripheral Control Registers, RAM and ROM are treated exactly the same. This makes the HPC easy to program. The HPC instruction set has relatively few opcodes but allows any of these opcodes to be used with any addressing mode so as to provide an Instruction Set with great power and flexibility.

Once the contents of this note have been understood, HPC code can be written without referring to any document more lengthy than the HPC Instruction Set description in the data sheet.

Section 6

MICROWIRE and MICROWIRE/PLUS Peripherals

Section 6 Contents

National Semiconductor

MICROWIRE™ and MICROWIRE/PLUS™: 3-Wire Serial Interface

National's MICROWIRE and MICROWIRE/PLUS provide for high-speed, serial communications in a simple 3-wire implementation.

Originally designed to interface COP400 microcontrollers to peripheral devices, the MICROWIRE protocol has been extended to both the COP800 and HPC™ families with the enhanced version, MICROWIRE/PLUS.

Because the shift clock in MICROWIRE/PLUS can be internal or external, the interface can be designated as either bus master or slave, giving it the flexibility necessary for distributed and multiprocessing applications.

With its simple 3-wire interface, MICROWIRE/PLUS can connect a variety of nodes in a serial-communication network.

This simple 3-wire design also helps increase system reliability while reducing system size and development time.

MICROWIRE/PLUS consists of an 8-bit serial shift register (SIO), serial data input (SI), serial data output (SO), and a serial shift clock (SK).

Because the COP800 and HPC families have memory-mapped architectures, the contents of the SIO register can be accessed through standard memory-addressing instructions.

The control register (CNTRL) is used to configure and control the mode and operation of the interface through user-selectable bits that program the internal shift rate. This greatly increases the flexibility of the interface.

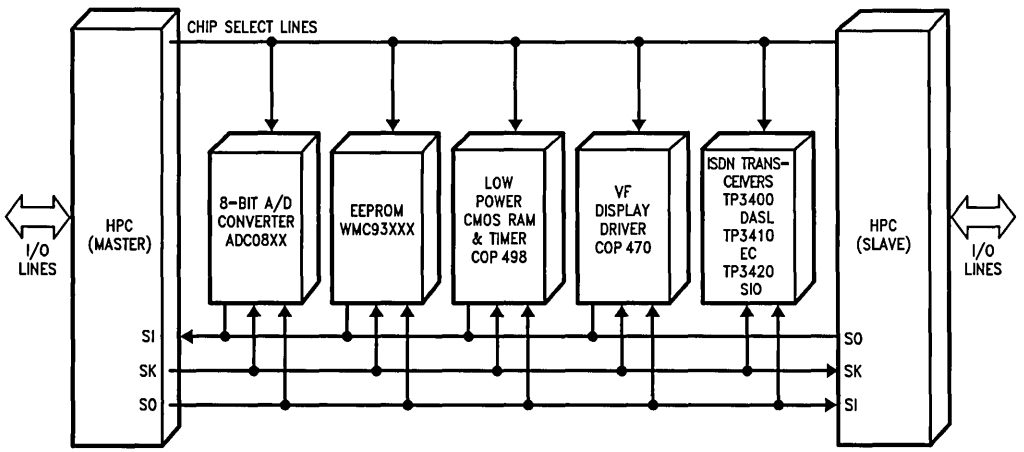
MICROWIRE/PLUS can also provide additional I/O capability for COP800 and HPC microcontrollers by connecting, for example, external 8-bit parallel-to-serial shift registers to 8-bit serial-to-parallel shift registers.

And it can interface a wide variety of peripherals:

- Memory (CMOS RAM and EEPROM)
- A/D converters
- Timers/counters
- Digital phase locked-loops
- Telecom peripherals
- Vacuum fluorescent display drivers
- LED display drivers
- LCD display drivers

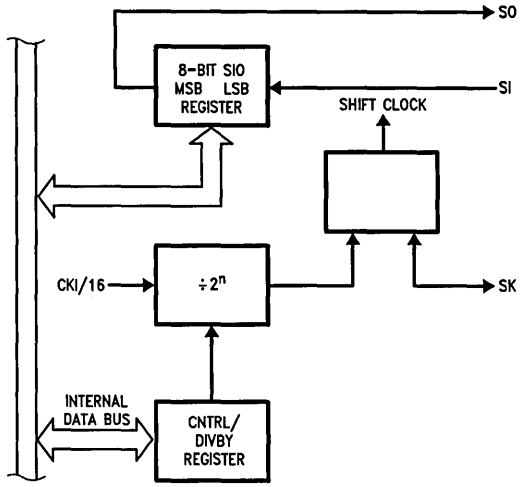
Both MICROWIRE and MICROWIRE/PLUS give all the members of National's microcontroller families the flexibility and design-ease to implement a solution quickly, simply, and cost-effectively.

MICROWIRE/PLUS System Block



TL/XX/0074–1

MICROWIRE/PLUS Block Diagram



TL/XX/0074-2

MICROWIRE and MICROWIRE/PLUS Peripherals

Part Number	Description	Databook
A/D CONVERTERS AND COMPARATORS		
ADC0811	11 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0819	19 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0831	1 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0838	8 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0832	2 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0833	4 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0834	4 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0852	Multiplexed Comparator with 8-Bit Reference Divider	Linear
ADC0854	Multiplexed Comparator with 8-Bit Reference Divider	Linear
DISPLAY DRIVERS		
COP470	4 Digit by 8 Segment Expandable V.F. Display Driver	Microcontroller
COP472-3	3 x 12 Multiplexed Expandable LCD Display Driver	Microcontroller
MM5450	35 Output LED Display Driver	Interface
MM5451	34 Output LED Display Driver	Interface
MM5483	31 Segment LCD Display Driver	Interface
MM5484	16 Segment LED Display Driver	Interface
MM5486	33 Output LED Display Driver	Interface
MM58201	8 Backplane and 24 Segment Multiplexed LCD Driver	Interface
MM58241	32 Output High Voltage Display Driver	Interface
MM58242	20 Output High Voltage Display Driver	Interface
MM58248	35 Output High Voltage Display Driver	Interface
MM58341	32 Output High Voltage Display Driver	Interface
MM58342	20 Output High Voltage Display Driver	Interface
MM58348	35 Output High Voltage Display Driver	Interface
MEMORY DEVICES		
COP498	4 x 64 Low Power CMOS RAM and Timer with "Wake-Up"	Microcontroller
COP499	4 x 64 Low Power CMOS RAM	Microcontroller
NMC9306	16 x 16 NMOS EEPROM	Memory
NMC9313B	16 x 16 NMOS EEPROM	Memory
NMC9314B	64 x 16 NMOS EEPROM	Memory
NMC9346	64 x 16 NMOS EEPROM	Memory
NMC93C06	16 x 16 CMOS EEPROM	Memory
NMC93C26	32 x 16 CMOS EEPROM	Memory
NMC93C46	64 x 16 CMOS EEPROM	Memory
NMC93C506	16 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C526	32 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C546	64 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C556	128 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C56	128 x 16 CMOS EEPROM	Memory
NMC93C566	256 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C66	256 x 16 CMOS EEPROM	Memory

MICROWIRE and MICROWIRE/PLUS Peripherals (Continued)

Part Number	Description	Databook
TELECOM DEVICES		
TP3400	Digital Adapter for Subscriber Loops (DASL)	Telecom
TP3410	Echo Canceller (EC)	Telecom
TP3420	S Interface Device (SID)	Telecom
AUDIO AND RADIO DEVICES		
DS8906	AM/FM Digital PLL Synthesizer	Interface
DS8907	AM/FM Digital PLL Frequency Synthesizer	Interface
DS8908	AM/FM Digital PLL Frequency Synthesizer	Interface
DS8911	AM/FM/TV Sound Up-Conversion Frequency Synthesizer	Interface
LMC1992	Stereo Volume/Tone/Fade with Source Select	Linear
LMC1993	Stereo Volume/Tone/Fade/Loudness with Source Select	Linear
LMC835	7 Band Graphic Equalizer	Linear
SPECIAL FUNCTIONS		
COP452L	Frequency Generator and Counter	Microcontroller

National Semiconductor

COP452L/COP352L Frequency Generator and Counter

General Description

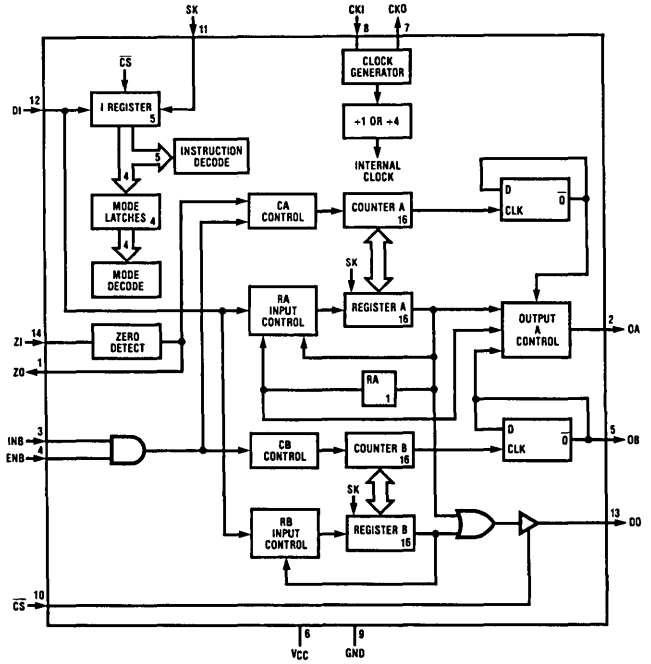
The COP452L and COP352L are peripheral members of the COPS™ family fabricated using N-channel silicon gate MOS technology. Containing two independent 16-bit counter/register pairs, they are well suited to a wide variety of tasks involving the measurement and/or generation of times and/or frequencies. Included in the features are multiple tone generation, precise duty cycle generation, event counting, waveform measurement, frequency bursts, delays, and "white noise" generation. An on-chip zero crossing detector can trigger a pulse with a programmed delay and duration. The COP352L is the extended temperature version of the COP452L. The COP352L is the functional equivalent of the COP452L.

The COP452L series peripheral devices can perform numerous functions that a microcontroller alone cannot perform. They can execute one or more complex tasks, attaining higher accuracies over a broader frequency range than a microcontroller alone. These devices remove repetitive yet demanding counting, timing, and frequency related functions from the microcontroller, thereby freeing it to perform other tasks or allowing the use of a simpler microcontroller in the system.

Features

- Unburdens microcontroller by performing "mundane" tasks
- Wider range and greater accuracy than microcontroller alone
- Generates frequencies, frequency bursts, and complex waveforms
- Measures waveform duty cycle
- Two independent pulse/event counters
- True zero crossing detector triggers output pulse
- White noise generator
- Compatible with all COP400 microcontrollers
- MICROWIRE™ compatible serial I/O
- 14-pin package
- Single supply operation (4.5V–6.3V, COP452L; 4.5V–5.5V, COP352L)
- Low cost
- TTL compatible

Block Diagram



TL/DD/6155–1

FIGURE 1

COP452L
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin (except ZI) relative to GND	−0.5V to +7.0V
Voltage at Pin ZI relative to GND	−0.8 to +10V
Sink Current, Output OA	15 mA
Sink Current, All Other Outputs	5 mA
Total Sink Current	35 mA
Source Current, Outputs OA, OB	5 mA
Source Current, All Other Outputs	1 mA
Total Source Current	10 mA
Ambient Operating Temperature	0°C to 70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.5W at 25°C 0.2W at 70°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A + 70°C$, $4.5V \le V_{CC} \le 6.3V$ (COP452L), unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})		4.5	6.3	V
Operating Supply Current	All Outputs Open		14	mA
Input Voltage Levels				
CKI Input Levels	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5.0V ±5%	2.0		V
Logic Low (V_{IL})			0.4	V
DI, INB, ENB, SK, $\overline{CS}$				
Logic High	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5.0V ±5%	2.0		V
Logic Low (V_{IL})			0.8	V
ZI Input Voltage		−0.8	+10	V
Impedance to GND at ZI		−1.6	7.8	kΩ
ZI Offset Voltage	(Note 1)		150	mV
Output Voltage Levels				
TTL Operation	V_{CC} = 5.0V ±5%			
Logic High (V_{OH})	I_{OH} = 100 μA	2.4		V
Logic Low (V_{OL})	I_{OL} = −1.6 mA		0.4	V
Maximum Allowable Output Current Levels				
Sink Current				
OA	(Note 2)		15	mA
All Other Outputs	(Note 2)		5.0	mA
Total Sink Current	(Note 3)		35	mA
Source Current				
OA, OB	(Note 2)		−5.0	mA
All Other Outputs	(Note 2)		−1.0	mA
Total Source Current	(Note 3)		−10	mA

Note 1: ZI offset voltage is the absolute value of the difference between the voltage at ZI and ground (pin 9) that will cause the zero detect circuit output to change state. This is the maximum value which takes into account the worst case effects of process, temperature, voltage, and gain variation.

Note 2: The maximum current for the specified pin must be limited to this value or less.

Note 3: The total current in the device must be limited to this value or less.

COP452L
AC Electrical Characteristics $0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise specified

Parameter		Conditions	Min	Max	Units
CKI Input Frequency (f_{IN})		÷4 Mode	100	2100	kHz
		÷1 Mode	64	525	kHz
Duty Cycle		÷4	30	55	%
÷1		÷1	45	55	%
Rise Time (t_r)		f_{IN} = 2.1 MHz		50	ns
Fall Time (t_f)		f_{IN} = 2.1 MHz		40	ns
SK Input Frequency			25	250	kHz
SK Duty Cycle			30	70	%
Internal Clock Frequency (f_I)			25	525	kHz
Internal Count Rate			0	$f_I/2$	Hz
Output Frequency			$f_I/131072$	$f_I/2$	Hz
Inputs					
DI	t_{SETUP}		800		ns
	t_{HOLD}		1.0		μs
Outputs					
CKO	t_{pd1}	C_L = 50 pF		0.2	μs
	t_{pd0}			0.2	μs
ZO	t_{pd1}	ZI = Sine Wave *(Figure 4)*		0.7	μs
	t_{pd0}			0.6	μs
DO	t_{pd1}	C_L = 50 pF		1.0	μs
	t_{pd0}			0.6	μs
OA	t_{pd1}	C_L = 50 pF V_{OUT} = 1.5V		0.7	μs
	t_{PDO}			0.8	μs
OB	t_{pd1}			1.0	μs
	t_{pd0}			0.4	μs

COP352L
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Any Pin (except ZI) relative to GND	−0.5V to +7.0V
Voltage at Pin ZI relative to GND	−0.8V to +10V
Sink Current, Output OA	15 mA
Sink Current, All Other Outputs	5 mA
Total Sink Current	35 mA
Source Current, Outputs OA, OB	5 mA
Source Current, All Other Outputs	1 mA
Total Source Current	10 mA
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.5W at 25°C 0.125W at 85°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le 85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})		4.5	5.5	V
Operating Supply Current	All Outputs Open		16	mA
Input Voltage Levels				
CKI Input Levels	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5.0V ±5%	2.2		V
Logic Low (V_{IL})			0.3	V
DI, INB, ENB, SK, $\overline{CS}$				
Logic High	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5.0V ±5%	2.2		V
Logic Low (V_{IL})			0.6	V
ZI Input Voltage		−0.8	+10	V
Impedance to GND at ZI		1.6	7.8	kΩ
ZI Offset Voltage	(Note 1)		150	mV
Output Voltage Levels				
TTL Operation	V_{CC} = 5.0V ±5%			
Logic High (V_{OH})	I_{OH} = 100 μA	2.4		V
Logic Low (V_{OL})	I_{OL} = −1.6 mA		0.4	V
Maximum Allowable Output Current Levels				
Sink Current				
OA	(Note 2)		15	mA
All Other Outputs	(Note 2)		5.0	mA
Total Sink Current	(Note 3)		35	mA
Source Current				
OA, OB	(Note 2)		−5.0	mA
All Other Outputs	(Note 2)		−1.0	mA
Total Source Current	(Note 3)		−10	mA

Note 1: ZI offset voltage is the absolute value of the difference between the voltage at ZI and ground (pin 9) that will cause the zero detect circuit output to change state. This is the maximum value which takes into account the worst case effects of process, temperature, voltage, and gain variation.

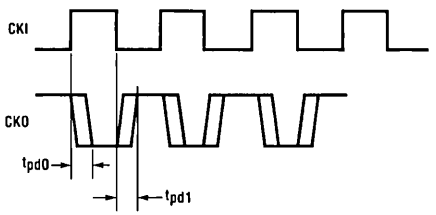
Note 2: The maximum current for the specified pin must be limited to this value or less.

Note 3: The total current in the device must be limited to this value or less.

COP352L
AC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise specified

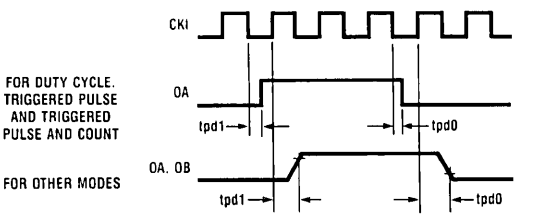
Parameter		Conditions	Min	Max	Units
CKI Input Frequency (f_{IN})		÷4 Mode	256	2100	kHz
		÷1 Mode	64	525	kHz
Duty Cycle		÷4	35	55	%
		÷1	50	55	%
Rise Time (t_r)		f_{IN} = 2.1 MHz		50	ns
Fall Time (t_f)		f_{IN} = 2.1 MHz		40	ns
SK Input Frequency			25	250	kHz
SK Duty Cycle			30	70	%
Internal Clock Frequency (f_I)			64	525	kHz
Internal Count Rate			0	$f_I/2$	Hz
Output Frequency			$f_I/131072$	$f_I/2$	Hz
Inputs					
DI	t_{SETUP}		800		ns
	t_{HOLD}		1.0		μs
Outputs					
CKO	t_{pd1}	C_L = 50 pF		0.25	μs
	t_{pd0}			0.25	μs
ZO	t_{pd1}	ZI = sine wave *(Figure 4)*		0.8	μs
	t_{pd0}			0.7	μs
DO	t_{pd1}	C_L = 50 pF		1.1	μs
	t_{pd0}			0.7	μs
OA	t_{pd1}	C_L = 50 pF, V_{OUT} = 1.5V		0.7	μs
	t_{pd0}			0.8	μs
OB	t_{pd1}			1.0	μs
	t_{pd0}			0.4	μs

Timing Diagrams



TL/DD/6155-2

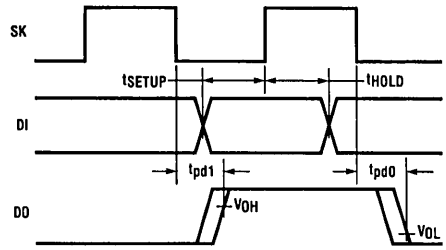
FIGURE 2a. CKO Output Timing



TL/DD/6155-3

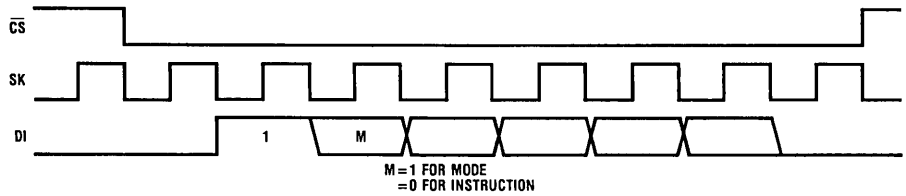
FIGURE 2b. OA and OB Output Timing

Timing Diagrams (Continued)



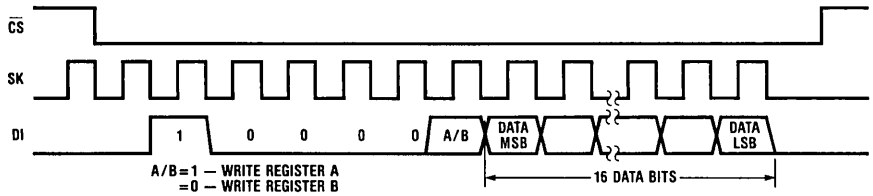
TL/DD/6155-4

FIGURE 3a. Synchronous Data Timing



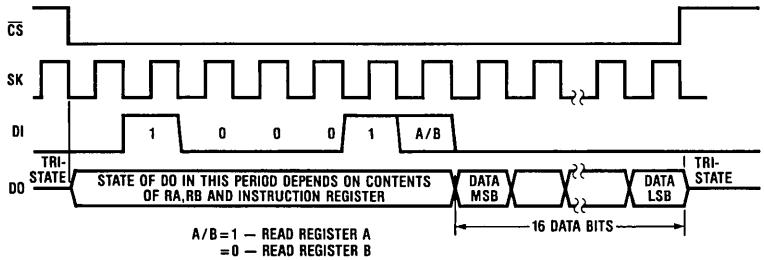
TL/DD/6155-5

FIGURE 3b. Instruction Timing (Except Read/Write)



TL/DD/6155-6

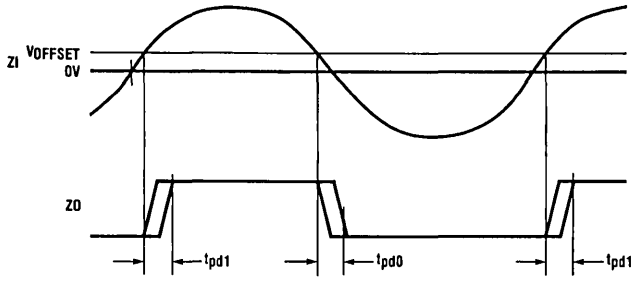
FIGURE 3c. Write Instruction Timing



TL/DD/6155-7

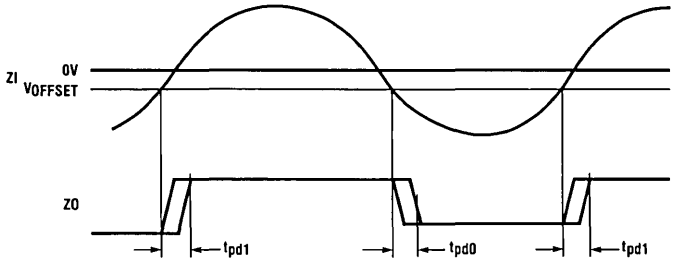
FIGURE 3d. Read Instruction Timing

Timing Diagrams (Continued)



TL/DD/6155-8

FIGURE 4a. ZO Timing, $V_{OFFSET} > 0V$



TL/DD/6155-9

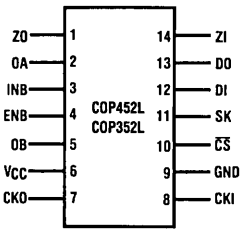
FIGURE 4b. ZO Timing, $V_{OFFSET} < 0V$

Pin Descriptions

Pin	Description	Pin	Description
ZO	Zero Cross Output Signal	CKI	Crystal Oscillator Input
OA	Counter A, Logic Controlled Output	GND	Ground
INB	Counter B, External Input	$\overline{CS}$	Chip Select
ENB	Enable for INB	SK	Serial Data I/O Clock Input
OB	Counter B Output	DI	Serial Data Input
V_{CC}	Power Supply	DO	Serial Data Output
CKO	Crystal Oscillator Output	ZI	AC Waveform Input, Counter A External Input

Connection Diagram

Dual-In-Line Package



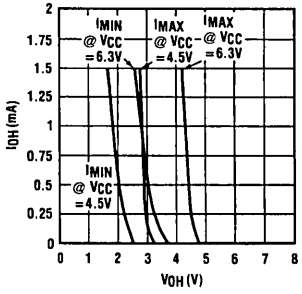
TL/DD/6155-10

FIGURE 5. Pin Connection Diagram

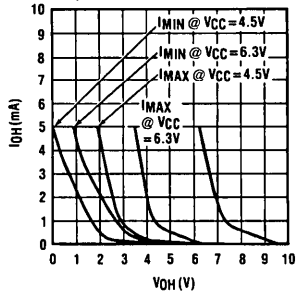
Order Number COP452D, COP352D, COP452N or COP352N
See NS Package Number D14D or N14A

Typical Performance Characteristics

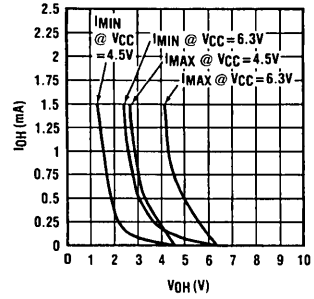
DO Source Current



OA, OB Source Current

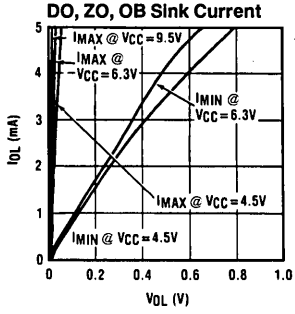


ZO Source Current

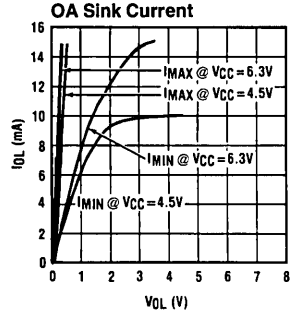


TL/DD/6155-11

DO, ZO, OB Sink Current



OA Sink Current

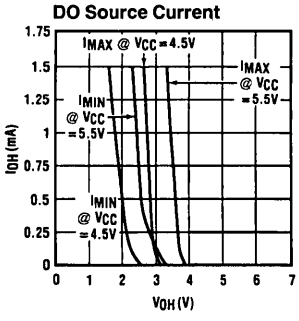


TL/DD/6155-12

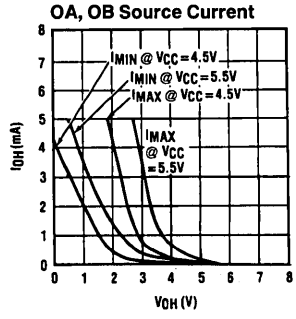
TL/DD/6155-13

FIGURE 6. COP452L

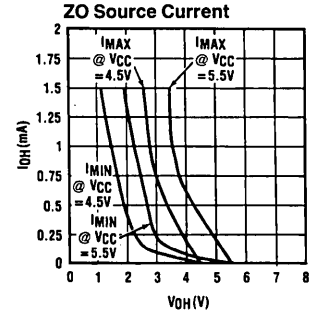
DO Source Current



OA, OB Source Current

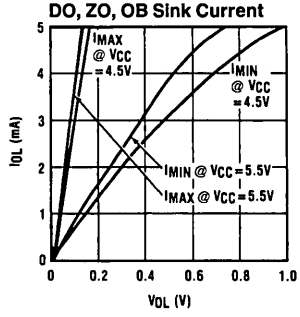


ZO Source Current

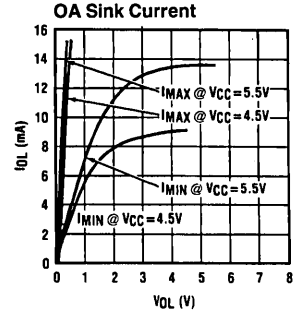


TL/DD/6155-14

DO, ZO, OB Sink Current



OA Sink Current



TL/DD/6155-15

TL/DD/6155-16

FIGURE 7. COP352L

Functional Description

The COP452L and COP352L are functionally identical devices. They differ only in V_{CC} range and/or operating temperature range, and certain electrical parameters associated with those temperature and voltage ranges. The following information will refer only to the COP452L. All the information, however, applies equally to the COP452L and COP352L.

INSTRUCTION SET AND OPERATING MODES

The COP452L has ten instructions and eleven operating modes as indicated in *Figure 8*. The information for the instruction or mode is sent to the COP452L via the serial interface. The MSB is always a "1" and is properly viewed as a start bit. The second MSB identifies the communication as an instruction or a mode. The lower four bits contain the command for the device.

Instruction	Opcode MSB LSB	Comments
LDRB	100000	Load register B from DI
LDRA	100001	Load register A from DI
RDRB	100010	Read register B to DO
RDRA	100011	Read register A to DO
TRCB	100100	Transfer register B to counter B
TRCA	100101	Transfer register A to counter A
TCRB	100110	Transfer counter B to register B
TCRA	100111	Transfer counter A to register A
CK1	101000	CKI divide by one
CK4	101001	CKI divide by four
LDM	11xxxx	Load mode latches

FIGURE 8a. COP452L Instruction Set

Operating Mode	Opcode MSB LSB
Reset	111111
Dual Frequency	110000
Frequency and Count	110100
Dual Count	110101
Number of Pulses	110010
Duty Cycle	110011
Waveform Measurement	110110
Triggered Pulse	110001
Triggered Pulse and Count	110111
White Noise and Frequency	111000
Gated White Noise	111001

FIGURE 8b. COP452L Operating Modes

A block diagram of the COP452L is given in *Figure 1*. Positive logic is used. The COP452L can execute ten instructions as indicated in *Figure 8a*, and has eleven operating modes. The operating mode is under user software control.

The device basically consists of two sixteen bit shift registers and two sixteen bit binary down counters organized as two register-counter pairs. In most operating modes, the two register-counter pairs are completely independent of one another. For frequency generation, both the register and counter of a given pair are utilized. The counter counts down to zero where a toggle flip-flop is toggled. Then the data in the register is loaded, automatically, to the counter and the process continues. A similar procedure is used in the duty cycle mode and number of pulses modes. For counting, the counters count the pulses at their respective inputs. There is no automatic counter-register transfer in the count modes. The counters wraparound from 0 to FFFF in the count modes. Data I/O is via the serial port and the registers. The counters are not involved in the input/output process at all.

The device requires a low chip select signal. When the device is selected ($\overline{CS}$ low) the driver on the DO pin is enabled and the device will accept data at DI on each SK pulse. When the device is deselected ($\overline{CS}$ high) the DO driver is TRI-STATE® and the I register is reset to 0. Note that chip select does not affect any other portion of the device. The mode latches are not affected. The COP452L will continue to operate in the mode specified by the user until the mode is changed by the user.

The COP452L contains a clock generator. The user may connect a crystal network to CKI and CKO or he may drive CKI from an external oscillator. Certain RC and LC networks may also be used. See the applications for further information.

The user also has control over whether the clock generator divides the CKI signal by 4 or 1. This allows the user to quickly get a 4 to 1 change in frequency output or input count rates. Alternatively, it allows the user to use a higher speed crystal or clock generator. The internal clock frequency (the frequency after the divider) must remain between the specified limits to guarantee proper operation. The state of the divider is not affected by $\overline{CS}$.

There is an internal power-on reset circuit which places the device in the Reset mode (mode latches all set to 1) and sets the clock divider to divide by four. If the CKI frequency is less than four times the minimum internal frequency the first access of the COP452L *must* be the command to set the divider to divide by 1. This command will be accepted and will be processed. Proper operation of the COP452L is not guaranteed if the internal frequency is less than the specified minimum. The power-on reset circuit does not affect the counter and registers of the COP452L.

When the COP452L is subjected to rapid power supply cycling, the internal power on reset will not function. Power must be removed for at least 20 seconds to allow restoration of internal reset circuitry. If the application requires power on-off cycles more frequently than once each 20 seconds the software reset with proper CKI divide by must be used to establish the initial state of the COP452L.

INSTRUCTION DESCRIPTION

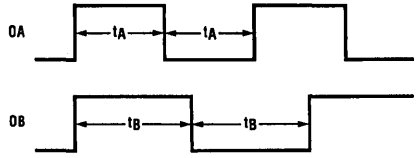
1. **Load Register (LDRA/LDRB)**—The selected register (A/B) is loaded with 16 bits of data shifted in on DI and clocked in by SK.
2. **Read Register (RDRA/RDRB)**—The data in the selected register (A/B) is shifted out serially onto DO. At the same time the data is recirculated back to the register.
3. **Load Counter (TRCA/TRCB)**—The contents of the selected register are transferred to its associated counter. (Counter A is loaded from register A; counter B is loaded from register B.) The contents of the register are unaffected.
4. **Copy Counter (TCRA/TCRB)**—The contents of the selected counter are transferred to its associated register. (Counter A loads register A; counter B loads register B.) The contents of the counter are unaffected.

Functional Description (Continued)

5. **CKI Divide by One**—The oscillator divider at the CKI input is set to divide by one. The internal frequency is therefore equal to the CKI frequency. This instruction should not be used if the CKI frequency is greater than the maximum internal frequency.
6. **CKI Divide by Four**—The oscillator divider at the CKI input is set to divide by four. The internal frequency is therefore equal to one-fourth of the CKI frequency. This instruction should not be used if the CKI frequency is less than four times the minimum internal frequency.
7. **Load Mode Latches**—The four mode latches are loaded with the lower four bits of the instruction.

MODE DESCRIPTION

1. **Reset Mode**—This mode sets OA and OB to "0". The mode latches are all set to "1". No counting occurs; the COP452L is in an idle condition. The registers and counters are not altered in any way.
2. **Dual Frequency**—Two frequencies are generated—one at output OA and one at output OB. The period of the square wave at OA is determined by the contents of register A. The period of the square wave at OB is determined by the contents of register B. In frequency generation modes, the counters count down until they reach zero. At that point the output toggles and the counters are automatically loaded from the respective registers. The counters are only loaded when they count down to zero. Therefore it may be necessary to initially load the counters. The frequency outputs at OA and OB are completely independent of one another. The respective counter inputs (INB, ZI) have no effect on the counters in this mode.



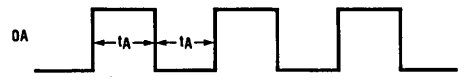
TL/DD/6155-17

$t_A = (A + 1)t$
$t_B = (B + 1)t$
$0 \le A \le 65535; 0 \le B \le 65535$

Where: A = Contents of register A
B = Contents of register B
t = Period of internal clock
= Period of CKI oscillator (÷ mode)
= 4 × period of CKI oscillator (÷4 mode)

Period of output square wave = 2(N + 1)t
Where t is devined above
N = Contents of register
$0 \le N \le 65535$ ($0 \le N \le FFFF_{16}$)

3. **Frequency and Count**—A single frequency is output at OA. Counter B counts external pulses on INB (when ENB = 1). There is no automatic clear of the counter. Since counter B counts down from whatever state it is in it is usually desirable to preload the counter. Preloading the counter with all zeroes will give the two's complement of the count. Preloading the counter with all ones will give the one's complement of the count.



TL/DD/6155-18

$t_A = (A + 1)t$

Where: A = Contents of register A
t = Period of internal clock
(as previously defined)
$0 \le A \le 65535$ ($0 \le A \le FFFF_{16}$)

OB toggles each time counter B counts through zero.

Maximum count rate at INB = $f_I/2$

Where: f_I = Internal Clock frequency
= CKI input frequency (÷1 mode)
= CKI input frequency ÷4 (÷4 mode)

Minimum pulse width required for reliable counting = t
where t = period of internal clock.

4. **Dual Count**—In this mode counter A and counter B are enabled as external event or pulse counters. Counter A counts pulses at ZI and counter B counts pulses at INB (when ENB = 1). There is no automatic clear of either counter. Each counter counts down from whatever state it starts in. Thus, to ease reading the information, the counters should be preloaded. Preloading the counters with all zeroes will give the two's complement of the count. Preloading the counters with all ones will give the one's complement of the count. The circuitry which decrements the counters is enabled by the high to low transition at the count input. There is no interaction between the two register counter pairs.

OA toggles every time counter A counts through "0".

OB toggles every time counter B counts through "0".

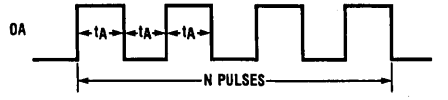
The counters, when counting, count down and wrap around from 0 to FFFF and continue counting down.

Maximum count rate = $f_I/2$
where: f_I = internal clock frequency

Minimum pulse width = t
where: t = period of internal clock
(as previously defined).

There is no requirement that the count signal be symmetrical. The pulse width low must be at least equal to t. The pulse width high must also be at least equal to t.

5. **Number of Pulses Mode**—This mode outputs at OA a specified number of pulses of a specified width. The number of pulses is specified by the contents of register B. The pulse width is specified by the contents of register A.



TL/DD/6155-19

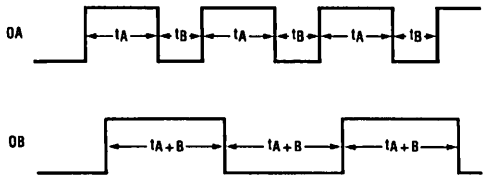
$t_A = (A + 1)t$
$N = B + 1$

Where: A = Contents of register A
B = Contents of register B
t = period of internal clock
(as previously defined)
$1 \le A \le 65535, A \ne 0$ ($1 \le A \le FFFF_{16}$)
$0 \le B \le 65535$ ($0 \le B \le FFFF_{16}$)

Functional Description (Continued)

OB toggles each time a pulse train is generated at OA. The pulse is generated each time the COP452L is selected and an instruction is set to the device. Counter B is automatically loaded from register B after the N pulses are generated. Counter A is automatically loaded from register A at each transition of OA. Therefore simply reloading the number of pulses mode will repeat the previous sequence.

6. Duty Cycle Mode—This mode generates a rectangular waveform at OA. The pulse width high is specified by the contents of register A. The pulse width low is specified by the contents of register B. A combination square wave signal is generated at OB.



TL/DD/6155-20

$t_A = At$

$t_B = Bt$

$t_{A+B} = (A + B)t$

Where: A = Contents of register A

B = Contents of register B

t = period of internal clock (as previously defined)

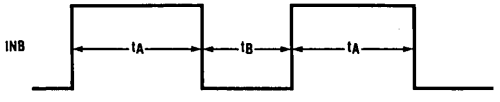
$1 \le A \le 65535, A \neq 0$ $(1 \le A \le FFFF_{16})$

$1 \le B \le 65535, B \neq 0$ $(1 \le B \le FFFF_{16})$

7. Waveform Measurement Mode—This mode measures the high and low times of an external waveform at INB (with ENB = 1). Counter A counts the pulse width high and counter B counts the pulse width low. On the high to low transition counter A is transferred to register A and then cleared. On the low to high transition counter B is transferred to register B and then cleared. The counters, therefore, count down from zero. Therefore the value read from the registers is a two's complement value. The transfer from the counter to register is inhibited during a read instruction.

The outputs OA and OB toggle each time the respective counter counts through zero.

The minimum pulse width, either high or low, that can be measured, is the period of the internal frequency. The maximum pulse width that can be measured is the maximum count (65535) multiplied by the period of the internal frequency.



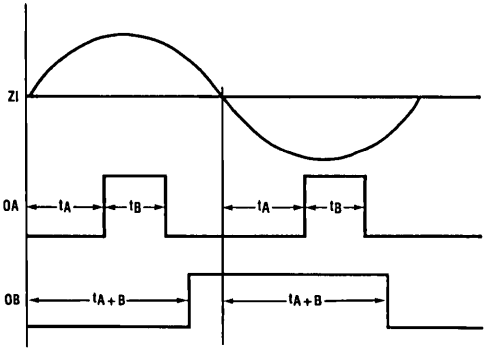
TL/DD/6155-21

$65535t \ge t_A \ge t$

$65535t \ge t_B \ge t$

Where: t = period of internal clock

8. Triggered Pulse Mode—This mode outputs a pulse triggered by the zero crossing of a signal at ZI. The delay from the zero crossing is specified by the contents of register A. The pulse width is specified by the contents of register B. Input INB is ignored. See applications section for further information.



TL/DD/6155-22

$t_A = (A + 1.5)t$

$t_B = Bt$

$t_{A+B} = (A + B + 1.5)t$

Where: A = Contents of register A

B = Contents of register B

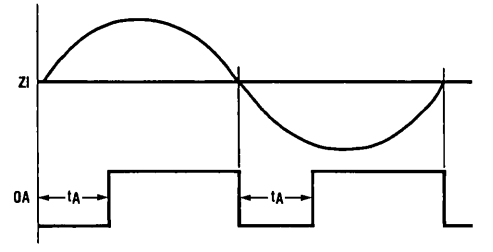
t = period of internal clock (as previously defined)

$0 \le A \le 65535$ $(0 \le A \le FFFF_{16})$

$1 \le B \le 65535, B \neq 0$ $(1 \le B \le FFFF_{16})$

9. Triggered Pulse and Count Mode—This mode outputs a pulse at OA triggered by the zero crossing of a signal at ZI. The contents of register A specify the delay from the zero crossing. The pulse remains high until the next zero crossing of the signal at ZI.

Independently of the zero detection, counter B counts external events at INB (when ENB = 1). The conditions on the counter as described previously apply here.



TL/DD/6155-23

$t_A = (A + 1.5)t$

Where: A = Contents of register A

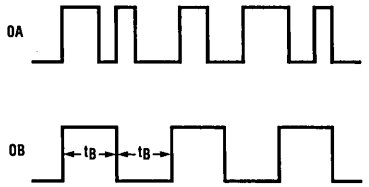
t = period of internal clock (as previously defined)

$0 \le A \le 65535$ $(0 \le A \le FFFF_{16})$

OB toggles each time counter B counts through 0

Functional Description (Continued)

10. White Noise and Frequency Mode—Register A is converted to a 17-stage shift register generator for the generation of pseudo-random noise at output OA. OB outputs a square wave whose period is specified by the contents of register B. The shift register generator is shifted at the internal frequency (= CKI frequency or ¼ CKI frequency depending on the oscillator divider). See the applications section for more information on the white noise generator.



TL/DD/6155-24

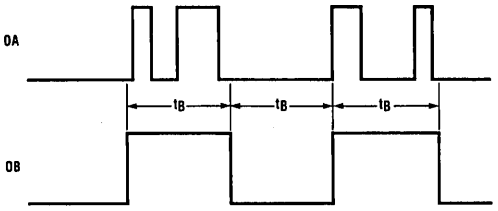
$t_B = (B + 1)t$

Where: B = Contents of register B

t = period of internal clock (as previously defined)

$0 \le B \le 65535$ $(0 \le B \le FFFF_{16})$

11. Gated White Noise Mode—This mode generates pseudo-random noise ANDed with a square wave. OA outputs this combined signal. OB outputs a square wave frequency. Register A is converted into a 17-stage shift register generator which is shifted at the internal frequency rate. Counter A is not used. Counter B and register B are used in the frequency generation. See the applications section for further information on the white noise generation.



TL/DD/6155-25

$t_B = (B + 1)t$

Where: B = Contents of register B

t = period of internal clock (as previously defined)

$0 \le B \le 65535$ $(0 \le B \le FFFF_{16})$

GENERAL NOTES

The master timing reference in the COP452L is the internal frequency. This is the CKI frequency after it has passed through the divider. This frequency must remain within its specified limits. The maximum count rate at either input is this frequency divided by 2. The minimum pulse width that can be measured is the period of this frequency.

$\overline{CS}$, other than removing DO from the TRI-STATE condition and allowing data to come into the I register via DI, does not affect the operation of the device. $\overline{CS}$ must go high between accesses in order to clear the I register. Since the I register is cleared when $\overline{CS}$ goes high, the user must insure that $\overline{CS}$ does not go high before the COP452L has accepted the information in the I register. See the software interface section for further explanation on this point. CS does not affect the mode latches.

In those modes where there is an automatic transfer from the register to the counter (frequency generation, duty cycle, number of pulses, triggered pulse), care must be exercised when reading or writing the register. To insure proper, "glitch-free" operation, one of the two procedures below must be followed:

1. Place the COP452L in the RESET mode.
2. Read or write the appropriate register.
3. Place the COP452L back in the original mode.

Alternatively:

1. Read or write the appropriate register.
2. Send the instruction to copy the appropriate register to its counter.

WARNING: Failure to observe one or the other of these procedures can cause some faulty output conditions.

The COP452L powers up in the RESET mode and with oscillator divide by 4. If the CKI input frequency is less than 4 times the minimum internal clock frequency the user *must* set the oscillator divider to divide by 1 *before* attempting any operation with the COP452L. The instruction setting the oscillator divider will be accepted regardless of the value of the internal clock frequency.

Caution: *Failure to observe this requirement will result in the improper operation of the COP452L.*

Applications Information

ZERO CROSS

The ZI input normally requires a resistor and diode external to the device as indicated in *Figure 9a*. The resistor is part of a voltage divider used to ensure that the voltage at pin ZI does not exceed 10V peak and to protect the diode which is required to clamp the negative voltage swing at the input to less than −0.8V. *Figure 9b* is the recommended input circuit if logic level pulses are input to ZI for counting.

As indicated above, the input voltage at ZI must not exceed 10V peak. For inputs less than 10V peak, the resistor in *Figure 9a* is required only to protect the diode. Otherwise, the resistor should be selected to guarantee that the voltage at pin ZI does not exceed 10V peak. *Figure 10* shows this resistor (R_S) and the impedance (R_{IN}) which forms the first part of the input circuit at ZI. The absolute value of R_{IN} can vary widely with process variation. The user should compute the divider with R_S and the worst case maximum of R_{IN} so that the voltage at pin ZI is 10V or less. The following relationship should be used when the input voltage is greater than 10V peak:

$$\frac{R_{IN(MAX.)}}{R_S + R_{IN(MAX.)}} \times V_{IN} \le 10V \text{ peak}$$

Substituting the maximum value for R_{IN} and solving for R_S gives:

$$R_S \le \frac{V_{IN}}{10} \times 7.8k - 7.8k$$

where: V_{IN} = peak input voltage.

Note that this equation is not valid for V_{IN} less than 10V. In this case, the value of R_S is chosen primarily for protection of the diode and not to divide the voltage down to acceptable values.

Applications Information (Continued)

ZERO CROSS OFFSET

As the electrical characteristics indicate, the ZI input has a worst case offset of 150 mV in the zero crossing detection. Therefore, the output of the zero cross detection circuit will change state within ±150 mV of zero volts. There are no directional characteristics to this, i.e., approaching zero from the positive or negative direction has no effect on where the output of the zero cross detection circuit will change state (see *Figure 4*). The offset further indicates that the voltage at pin ZI must exceed 150 mV peak in order to guarantee that the zero crossings will be detected and the appropriate signals generated.

TRIGGERED PULSE MODES

The delays from the zero crossing in the triggered pulse modes are measured from the point where the output of the zero crossing detection circuit changes state—the trip point of this circuit. As stated before, the delay time from this trip point is:

$$T = (A + 1.5)t$$

where: T = delay time from trip point

A = contents of register A

t = period of internal clock

The delay from the true zero crossing of the input waveform has other parameters that must be considered. The equation is of the form:

$$T = (A + 1.5)t \pm |X_1| + X_2 + X_3$$

where: T, A, t are as defined previously

X_1 = time for input waveform to reach the trip point of the zero cross detection circuit

X_2 = propagation delay through the zero cross detection circuit

X_3 = input synchronization delay

Parameter X_1 is dependent on the peak voltage at pin ZI and on the frequency of the input signal. The peak voltage at ZI is in turn dependent on the R_S–R_{IN} voltage divider and the input voltage. The X_1 time is added or subtracted because the trip point of the zero cross detection circuit may be either above or below zero. In the worst case, the trip point is the maximum offset of 150 mV. For a sine wave signal, X_1 is determined as follows:

$$V_{OFFSET} = V_p \sin[2\pi f(X_1)]$$

$$X_1 = \frac{1}{2\pi f} \arcsin \frac{V_{OFFSET}}{V_P}$$

and

$$V_P = V_{IN} \frac{R_{IN}}{R_S + R_{ON}}$$

substituting we have

$$X_1 = \frac{1}{2\pi f} \arcsin \left(V_{OFFSET} \frac{R_S + R_{IN}}{V_{IN} R_{IN}} \right)$$

where: V_{OFFSET} = zero crossing offset or trip point

V_P = peak input voltage at pin ZI

f = frequency of input signal

R_{IN} = internal impedance to ground at pin ZI

R_S = external series resistance at ZI

Both V_{OFFSET} and R_{IN} vary from device to device. It is clear from the equation above that the maximum value of $|X_1|$ is obtained when V_{OFFSET} is at its maximum of 150 mV and R_{IN} is at its minimum of 2.6 kΩ. The minimum value of $|X_1|$ is obtained if V_{OFFSET} is 0. Using this information, the following range of $|X_1|$ is obtained:

$$0 \le |X_1| \le \frac{1}{2\pi f} \arcsin 0.15 \frac{R_S + 2.6k}{V_{IN} \times 2.6k}$$

Parameter X_2 is the propagation delay through the zero crossing detection circuit and its range is given by:

$$0.3\ \mu s \le X_2 \le 0.6\ \mu s$$

Parameter X_3 is the internal synchronization delay and is dependent upon when the zero crossing occurs relative to the internal timing which reads the output of the zero crossing detection circuit. The range for X_3 is:

$$0 \le X_3 \le \frac{t}{2}$$

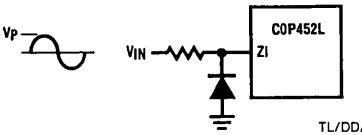
where: t = period of internal clock

With the preceding information, minimum and maximum values of the delay from true zero can be derived by simply substituting into the original equation.

$$T_{MIN} = (A+1.5)t - \frac{1}{2\pi f} \arcsin \left(0.15 \frac{R_S + 2.6k}{V_{IN} \times 2.6k} \right) + 0.3\ \mu s$$

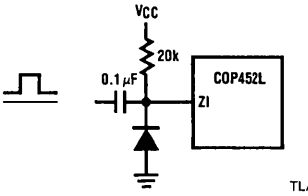
$$T_{MAX} = (A+1.50)t + \frac{1}{2\pi f} \arcsin \left(0.15 \frac{R_S + 2.6k}{V_{IN} \times 2.6k} \right) + 0.6\ \mu s + \frac{t}{2}$$

The preceding information should enable the user to determine more closely the actual delay from zero of output OA of the COP452L. This analysis applies to both of the triggered pulse modes. The three parameters, X_1, X_2, X_3, also apply in the same way in the triggered pulse and count mode when OA returns to 0 since it is the zero cross detection circuit that causes the output to return to zero in that mode.



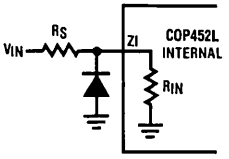
TL/DD/6155–26

FIGURE 9a



TL/DD/6155–27

FIGURE 9b



TL/DD/6155–28

FIGURE 10

Applications Information (Continued)

TRIGGERED PULSE MODES: INTERVENING ZERO CROSSINGS

In the triggered pulse modes, it is possible to specify a delay from the zero crossing which will extend beyond the next zero crossing. In the triggered pulse and count mode, the intervening zero crossing is ignored and therefore lost. The device will still continue to operate properly. The situation is somewhat different in the "pure" triggered pulse mode where both a delay and a pulse width are specified. Any zero crossing which occurs during the programmed delay time is ignored and therefore lost. However, if the delay time is counted out and the zero crossing occurs during the pulse width high time, the zero crossing will be recognized and the delay time will start counting again while the pulse width high time is being counted. This can result in a variety of possible conditions at the output—ranging from the apparent loss of that zero crossing to an effective very short delay from the zero crossing. What will occur depends on the values of the two counters and on their relationship to the times between zero crossings. Some interesting output waveforms can be produced, but their utility is questionable. Therefore, the user should exercise extreme caution in this mode and make sure that the times are such that all zero crossings occur at the "right" times. Otherwise, the user must be prepared to accept the bizarre effects that this situation can produce.

COUNT MODES

As stated before, the counters are 16-bit down counters. Preloading them when they are enabled as external event counters with ones or zeroes will give the one's or two's complement of the count. To read the counters it is necessary to first copy the counter to its respective register and then read the register.

The user can utilize the fact that the outputs toggle when the counter counts through zero. The counter can be preloaded with a value that represents the number of events the user wishes to count. When the output corresponding to that counter toggles, the specified number of events have occurred. Thus, the user can know that the required number of events have occurred without having to actually read the counter.

The counters require a pulse width greater than or equal to the period of the internal frequency in order to be reliably decremented. It is possible for a narrower pulse to decrement the counter, but it is not guaranteed. A narrower pulse will decrement the counter if it appears at the count input at the right time relative to the internal timing of the device. Since the user does not have access to this internal timing, it is impossible for him to synchronize the count input to this timing and effectively reduce the required width of the count pulse. Therefore, applying pulses at the count input of less than one period of the internal frequency in width may cause erratic counting in the sense that some of the pulses may be recognized and some may not be recognized. Reliable counting is assured only if the width of the count pulse is greater than or equal to one period of the internal frequency.

The counters decrement on a low-going pulse at the input. As stated above, the pulse must remain low at least one internal frequency period to give reliable counting. Similarly, the count signal must go high and remain high at least one internal frequency period before it goes low again. However, the count signal does *not* have to be symmetrical.

COP452L OSCILLATOR

The COP452L will operate over a wide range of oscillator input frequencies. The input frequency may be supplied from an external source or CKI and CKO can be used with a crystal or resonator to generate the oscillator frequency. *Figure 11* indicates some crystal networks for some typical crystal values.

RC and LC networks can also be connected between CKI and CKO to produce the oscillation frequency. *Figure 12* indicates some examples of such networks. *Figure 12a* is the recommended RC network for use in this manner. With C_1 = 0.005 μF, R = 1.5 kΩ, and C_2 between 20 pF and 400 pF oscillation frequencies between about 1 MHz and 2 MHz should be obtainable. The oscillation frequency decreases with increasing values of C_2. The user should feel free to experiment with the R and C values, and with the network configuration, to produce the oscillation frequency desired.

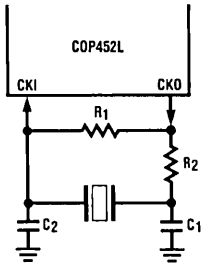
Figures 12b and *12c* indicate LC networks that can be used to produce the COP452L oscillation frequency. In *Figure 12b*, with L = 100 μH and C = 100 pF, a frequency of about 2 MHz should be produced. In *Figure 12c,* with L = 56 μH, C_2 = 27 pF, and C_1 between 33 pF and 0.01 μF, frequencies between about 1.5 MHz and 2 MHz can be produced.

There is, in effect, an inverter between CKI and CKO. This inverter was designed for use with a crystal and its associated network. It was not designed for use with the RC and LC networks previously described. However, these networks will work and are usable. The user should be prepared to experiment with the networks to determine component values, stability, oscillation frequency, etc. These networks should be viewed as the starting point for a user who wishes to use networks of this type to generate the COP452L oscillation frequency.

The RC networks provide an inexpensive way to generate the oscillation frequency. It is foolish, however, to expect any significant degree of frequency stability or accuracy over temperature and voltage with a simple RC network—especially if inexpensive, uncompensated components are used. LC and RLC networks can produce very stable and accurate frequencies. Regardless of the network used, the user must consider the variation of the external components in his design if accuracy and stability are important considerations in his application.

The crystal networks of *Figure 11* provide frequency stability and accuracy and are easy to use. If the application requires oscillation frequency accuracy and stability the crystal networks are recommended as the best solution.

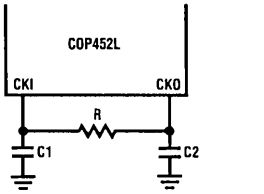
Applications Information (Continued)



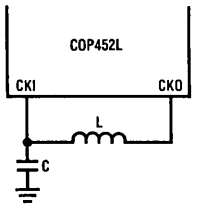
TL/DD/6155-30

Crystal Value	Component Values			
	R_1	R_2	C_1	C_2
455 kHz	1M	16k	80 pF	80 pF
32 kHz	1M	220k	6 pF–36 pF	30 pF

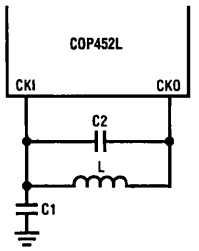
FIGURE 11. COP452 Crystal Oscillator



TL/DD/6155-31



TL/DD/6155-32



TL/DD/6155-33

a. b. c.

FIGURE 12. RC and LC Networks to Produce COP452 Oscillator Frequency

WHITE NOISE GENERATION MODES

In the two white noise modes register A is converted into a 17-stage shift register, or polynomial, generator. With feedback taps at stages 17 and 14, as indicated in *Figure 13*, a maximal length sequence is generated. With these feedback taps the characteristic polynomial of the sequence is:

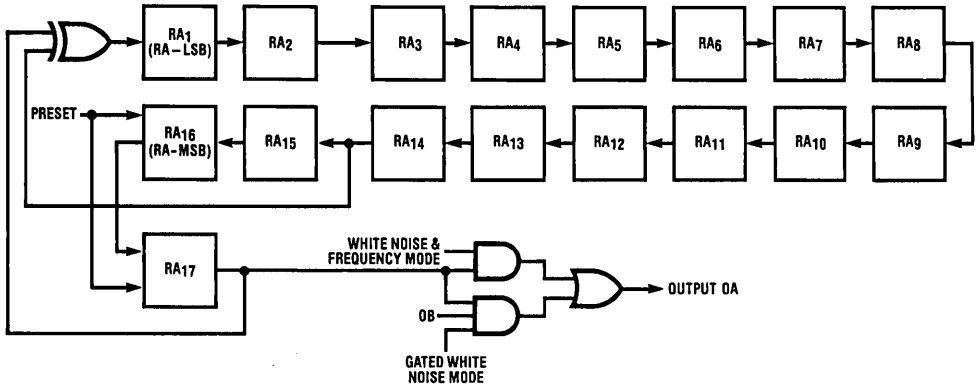
$$X^{17} + X^3 + 1.$$

The output of this generator is a pseudo-random sequence. Since the register is shifted at the internal frequency rate, the sequence repeats after a period equal to $(2^{17} - 1)t$, where t is the period of the internal frequency.

The first 16 stages of the shift register are the 16 bits of register A that the user may read or write. Entering either white noise mode presets the 16th stage to a 1 and connects the 17th stage to the shift register. If the user wishes, he can write register A and then enter the white noise and frequency mode. The output at OA will then be "1", and the lower 15 bits of the data user had written to register A. Following that, the polynomial sequence dictates the output. This injection of a 1 into the 16th stage prevents the lockup condition that occurs if all the stages are 0.

WARNING: To insure proper operation, the white noise must be entered from the Reset mode. The COP452 must be in the Reset mode before the desired white noise mode and there may be no intervening modes between Reset and the desired white noise mode. (The state of 17th stage is don't care (unknown).)

Applications Information (Continued)



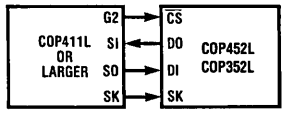
TL/DD/6155-34

Note: Setting the Register A to all 1's will result in a predictable pattern each time this mode is activated.

FIGURE 13. COP452L White Noise Generator

INTERFACE TO COPS MICROCONTROLLERS

Figure 14 indicates the typical interface between the COP452L and a COPS microcontroller. As is obvious from the figure, the interface is the standard MICROWIRE. G_2 is indicated as the chip select line because it is available on all COPS microcontrollers. Obviously, any convenient output of the microcontroller may be used as the chip select for the COP452L.



TL/DD/6155-35

FIGURE 14

The $\overline{CS}$ pin of the COP452L must be toggled between successive communications with the device. The internal I register (instruction register) is held reset (all zero) when $\overline{CS}$ is high. Since this is the only way in which the I register is cleared, failure to take $\overline{CS}$ high between accesses will result in improper operation.

The COP452L contains an internal power-on reset circuit which sets the mode latches to one, i.e., places the COP452L in the RESET mode, and sets the oscillator divider to divide by 4. The counters and registers are not affected by this reset circuit and are therefore undefined at power up.

INTERFACE SOFTWARE FOR THE COP452L

Sample software for interfacing COPS microcontrollers to the COP452L is given below. The code is completely general and will work in any COPS microcontroller. The following assumptions are made:

1. Pin G_2 is used as the chip select for the COP452L (because G_2 is available on all COPS microcontrollers).
2. G_2 is assumed high on entry to the routines.
3. The SK clock is off (0) on entry to the routines.
4. Register 0 of the microcontroller is arbitrarily chosen as the I/O register.
5. The leading digit sent out is of the form 001X where 1 is a start bit; X is 1 or 0, depending on the operation.
6. The next lower digit contains the remaining 4 bits of the command.
7. If data is being sent, it is in the next 16 bits of information sent.
8. Location GSTATE chosen as RAM address 0,15.
9. SK frequency is less than or equal to the internal frequency.

Since the COP452L is an I/O device, the code takes precautions to insure that SO is 0 prior to enabling the SK clock. (This is a wise precaution to take in any system with I/O peripherals on the serial port.)

Two versions of the WRITE routine are provided. The destructive WRITE routine destroys the information in the microcontroller as the data is being sent out to the COP452L. The nondestructive WRITE routine preserves the data in the microcontroller as that data is being sent out to the COP452L. The destructive routine is a little more code efficient than the nondestructive routine.

Applications Information (Continued)

```
WRCMND:   CLRA                ; SET UP POINTER FOR COMMAND ONLY WRITE
          AISC    1
          JP      WRITE
WRDATA:   CLRA                ; SET UP POINTER FOR COMMAND AND DATA WRITE
          AISC    5
WRITE:    LBI     GSTATE      ; GSTATE = LOCATION 0,15
          RMB     2
          OMG                 ; SEND COP452L CHIP SELECT LOW
          CAB                 ; POINT TO PROPER LOCATION FOR OUTPUT
          LEI     8           ; ENABLE SHIFT REGISTER MODE
          RC                  ; JUST TO INSURE SO = 0 BEFORE CLOCK ON
          CLRA
          XAS                 ; THESE 3 WORDS FOR SAFETY ONLY
          SC                  ; SO SK WILL TURN ON AT NEXT XAS
SEND:     LD
          XAS
          XDS
          JP      SEND
FINISH:   RC                  ; ALL DONE, SK OFF, DESELECT COP452L, AND SET
          XAS                 ; SO TO ZERO
DONE:     LBI     GSTATE
          SMB     2
          OMG
          LEI     0
          RET
;
CODE TO WRITE COP452L — DATA DESTROYED IN MICROCONTROLLER
```

TL/DD/6155–36

Applications Information (Continued)

The code below is the code to read COP452L. It is written so that the command to the COP452L is sent out nondestructively, i.e., the data in the microcontroller is preserved. A routine which sends out the data destructively could be easily generated but is not shown here. The user is referred to the techniques in the WRITE routines to determine how to modify this READ routine to send the command out destructively.

```
READ:    CLRA              ; READ INSTRUCTION IN 0, 1 AND 0, 0 AND IS
         AISC    1         ; OF THE FORM 00100010 OR 00100011 IF READ
         LBI     GSTATE    ; RA OR RB
         RMB     2
         OMG               ; SELECT THE COP452L
         CAB
         SC
         CLRA              ; SO THAT ZEROES GO OUT FIRST
         LEI     8
SEND2:   XAS
         LD
         XDS
         JP      SEND2     ; NONDESTRUCTIVE SENDING OF READ INSTRUCTION
         XAS
         CLRA              ; SET UP TO READ
         AISC    2
         CAB
         NOP               ; NOW WAIT FOR THE DATA
         NOP
         NOP
RDLOOP:  CLRA
         XAS
         XDS
         JP      RDLOOP
         RC                ; TURN OFF THE CLOCK
         XAS               ; READ LAST 4 BITS
         JP      DONE      ; COMMON EXIT WITH WRITE ROUTINE
                           ; EXITS WITH DATA IN LOWER 3 DIGITS OF RO
                           ; AND IN THE ACCUMULATOR
```

SAMPLE CODE TO READ THE COP452L

```
WRCMND:  CLRA              ; SET UP POINTER FOR COMMAND ONLY WRITE
         AISC    1
         JP      WRITE
WRDATA:  CLRA              ; SET UP POINTER FOR COMMAND AND DATA WRITE
         AISC    5
WRITE:   LBI     GSTATE
         RMB     2
         OMG               ; SELECT THE COP452L — G2 LOW
         CAB               ; LOAD THE POINTER
         RC
         CLRA
         LEI     8         ; ENABLE SHIFT REGISTER MODE
         XAS               ; SEND OUT ZEROES
         SC
         CLRA
SEND:    XAS               ; FIRST TIME THROUGH, TURNS ON CLOCK
         LD                ; THEN SENDS DATA
         XDS
         JP      SEND
         XAS               ; SEND LAST 4 BITS
         CLRA
         NOP
FINISH:  RC
         XAS               ; ALL DONE, SK OFF
DONE:    LBI     GSTATE
         SMB     2         ; DESELECT THE COP452
         OMG
         LEI     0         ; SEND SO LOW
         RET
```

CODE TO WRITE COP452L — DATA PRESERVED IN MICROCONTROLLER

TL/DD/6155–37

Applications Information (Continued)

The software interface routines provided above are general purpose routines written to work in the general case for all COPS microcontrollers. They are written as subroutines to be called by the main program. There is no question that other routines can be written to perform the required function. It is also clear that these routines can be reduced in specific applications. These routines should be viewed as providing a framework from which the user can develop routines which are optimal to a specific application.

Assumption 9 mentioned prior to the code itself presents an important requirement for the interface software. There must be a time delay greater than 3 periods of the internal frequency between the time the SK clock is turned off and the time the COP452L is deselected. This is required because the COP452L reads the instruction register with timing based on its internal frequency. When the microcontroller deselects the COP452L, CS goes high and the instruction register is automatically cleared. Therefore, depending on the relative speeds of SK and the internal frequency, it is possible that the instruction register may be cleared before the COP452L has accepted the information. The sample code provided automatically satisfies the requirement mentioned above whenever the SK frequency is less than or equal to the counter clock frequency. When SK is faster than the internal frequency, some delay may be required between the time SK is turned off and the time the COP452L is deselected. The time delay is not required when reading or writing the COP452L registers or when changing the oscillator divider.

Caution: *Failure to observe this time delay will result in improper operation of the COP452L.*

APPLICATION # 1—GENERATION OF MULTIPLE TONES

The COP452L makes the generation of two independent frequencies a simple task. This application indicates how to generate frequencies with the COP452L and also indicates other aspects of control of the device.

The requirement is to generate the following two DTMF frequencies:

$$f1 = 941\ \text{Hz}$$

$$f2 = 1336\ \text{Hz}$$

We will select the CKI frequency of the COP452L as 525 kHz. Therefore, in divide by 1 mode, the internal frequency is 525 kHz. Since the registers in the COP452L are loaded with a number related to the period of the frequency, we need the periods of f1 and f2.

$$\frac{1}{f1} = t1 = 1062.7\ \mu s; \quad \frac{t1}{2} = 531.35\ \mu s$$

$$\frac{1}{f2} = t2 = 748.5\ \mu s; \quad \frac{t2}{2} = 374.25\ \mu s$$

As stated earlier, the period of an output frequency in the COP452L in the frequency generation mode is given by:

$$T = 2(N + 1)t$$

where:

t = period of internal clock

N = register value

Solving for N, the equation becomes:

$$N = \frac{T}{2t} - 1$$

With the internal frequency at 1 MHz, the value of t is 1 μs. Therefore, the N values with which the registers must be loaded to generate the frequencies specified above are 278 (116 hex) and 195 (0C3 hex). Note that the fractional parts of the numbers are lost since the COP452L cannot be loaded with fractional numbers. Note that the fractionial parts may be reduced or eliminated by judicious choice of the CKI frequency. With the numbers here, the COP452L will generate a frequency with a period of 1062 μs (941.62 Hz) and a frequency with a period of 748 μs (1336.9 Hz). Note that these values are accurate to within 0.7% of the desired output frequencies.

Figure 15 indicates a connection diagram for this application. The software to accomplish this task is indicated below. The software indicates several aspects of the usage of the COP452L. The code first resets the COP452L, then loads the registers with the proper values, transfers the registers to the counters, puts the COP452L in the CKI divide by 1 state, and then loads the dual frequency mode. The output frequency generation begins when the dual frequency mode is loaded. The code as written is independent of the COP microcontroller used. The code uses the WRITE routines as described in the software interface section and assumes that these routines are located in the subroutine page.

```
            . PAGE    0
GSTATE      =         0, 15
POWUP:      CLRA
            XAS                   ; TURN OFF SK CLOCK (C = 0 AT POWER UP)
            LBI       GSTATE
            STII      15
            LBI       GSTATE
            OMG                   ; MAKE SURE COP452 IS DESELECTED
            LBI       0, 0
            JSRP      CLEAR       ; CLEAR REGISTER 0
            LBI       0, 0        ; NOW SET UP TO SEND RESET MODE TO COP452
            STII      15
            STII      3           ; RESET COMMAND AND START BIT
            JSRP      WRCMND
```

TL/DD/6155–38

Applications Information (Continued)

```
; THE COP452L IS NOW RESET, NOW SET UP TO WRITE REGISTER A TO
; GENERATE OUTPUT FREQUENCY OF 941 HZ AT OA

                LBI     0, 0
                STII    6               ; 116  HEX  =  278, GIVE PERIOD OF 1062 µs
                STII    1
                STII    1
                STII    0
                STII    1
                STII    2               ; START BIT PLUS CODE TO WRITE RA
                JSRP    WRDATA
; REGISTER A IS NOW LOADED. NEXT TRANSFER REGISTER A TO COUNTER A
                LBI     0, 0
                STII    5
                STII    2               ; INSTRUCTION TO TRANSFER PLUS START BIT
                JSRP    WRCMND
; ALL DONE WITH REGISTER AND COUNTER A, NEXT WORK ON REGISTER B
                LBI     0, 0
                STII    3               ; WRITE REGISTER B WITH 0C3 HEX (195)
                STII    C               ; TO GIVE FREQUENCY OF 1336 HZ
                STII    0
                STII    0
                STII    0               ; INSTRUCTION TO WRITE RB
                STII    2
                JSRP    WRDATA
; REGISTER B IS NOW LOADED. NEXT TRANSFER RB TO CB
                LBI     0, 0
                STII    4               ; INSTRUCTION TO TRANSER RB TO CB
                STII    2
                JSRP    WRCMND
; NOW LOAD CKI DIVIDE BY 1
                LIB     0, 0
                STII    8
                STII    2
                JSRP    WRCMND
; NOW PUT THE COP452 IN DUAL FREQUENCY MODE
                LBI     0, 0
                STII    0
                STII    3
                JSRP    WRCMND
; NOW THE CODE MAY PROCEED TO DO WHATEVER ELSE IS REQUIRED IN
; THE APPLICATION.
; THE SUBROUTINES USED IN THIS APPLICATION ARE CLEAR AND THE
; WRITE ROUTINES. THE ADD ROUTINE IS USED IN THE EXAMPLE BELOW
                . PAGE  2
CLEAR:          CLRA
                XIS
                JP      CLEAR
                RET
ADD:            SC
                LBI     2, 9            ; ROUTINE ADDS 1 TO COUNTER
ADD1:           CLRA
                ASC
                NOP
                XIS
                JP      ADD1
                RET
;
WRCMND:                                 ; SEE SOFTWARE INTERFACE FOR THIS ROUTINE
;
WRDATA:                                 ; SEE SOFTWARE INTERFACE FOR THIS ROUTINE
```

TL/DD/6155-39

Applications Information (Continued)

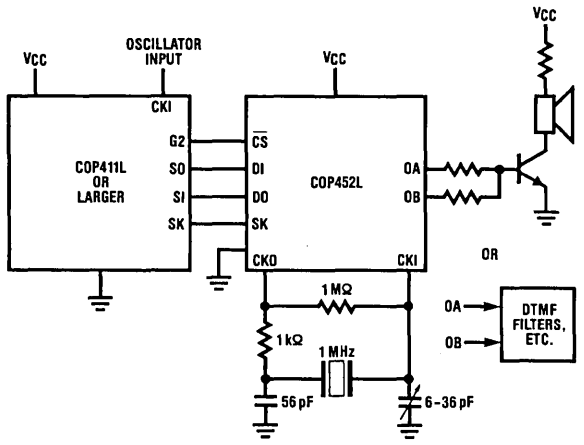
The preceding has done a lot with the COP452L. It is clear that the code can be reduced and specialized. The purpose here was to illustrate the various communications with the device.

An interesting effect can now be produced by making use of the 4 to 1 CKI divider. With the CKI frequency at 525 kHz, the internal frequency is well within the specified limits in either the divide by 1 or divide by 4 condition. Therefore, this characteristic of the device can be used to quickly multiply or divide the output frequency by 4. An interesting siren effect can thus be created. Sample code to do this is given below. This code assumes that the registers have been loaded and that the COP452 is in dual frequency mode. Again, the code is written to be independent of the COPS microcontroller used.

As is obvious from this code, it is a simple matter to create this effect. As was mentioned earlier, the code here is general purpose. This necessarily means that it can be reduced in specific applications. The user should view this code as representative of the techniques involved and then optimize or rewrite the routines to suit his particular application.

```
SIREN:   LBI     2, 9      ; USE REGISTER 2 AS COUNTER FOR DELAY TIME
         JSRP    CLEAR
         LBI     0, 0
         STII    8         ; CKI DIVIDE BY 1
         STII    2
         JSRP    WRCMND
PLUS1:   JSRP    ADD       ; INCREMENT COUNTER FOR DELAY
         SKC
         JP      PLUS1     ; EXIST DELAY LOOP WHEN COUNTER OVERFLOWS
         LBI     0, 0
         STII    9         ; CKI DIVIDE BY 4
         STII    2
         JSRP    WRCMND
         LBI     2,9
         JSRP    CLEAR
PLUS1A:  JSRP    ADD
         SKC               ; AGAIN, TIME OUT VIA THE COUNTER
         JP      PLUS1A
         JP      SIREN     ; DONE, START OVER AGAIN
```

TL/DD/6155-40



TL/DD/6155-41

FIGURE 15. Dual Frequency Application

Applications Information (Continued)

APPLICATION #2

This application makes use of the number of pulses mode of the COP452L to control a stepping motor. The technique is equally applicable in any situation where a number of pulses must be generated based upon the state of the system. *Figure 16* indicates the system interconnect. Since the oscillator frequency is 2.1 MHz max. and the CKO pin of the COP452L is being used to drive the CKI of the microcontroller, a COP420 is specified as the microcontroller. If a separate oscillator were provided, any COPS microcontroller could be used. The software is completely general and will work in any COPS microcontroller.

The application has the following specifications:

1. The pulse width required for the stepping motor is 5 ms ±5%.
2. The system has 4 return lines which indicate 4 possible variations in the number of output pulses required. These four conditions are:
   a. 10 pulses required
   b. 100 pulses required
   c. Repeat the last number of pulses sent
   d. Send one more than the last number of pulses
3. The system has a signal available indicating that the return lines contain valid information.
4. One pulse is required at power up.

A flowchart to implement this system is indicated in *Figure 17*. *Figure 16* is the interconnect used in this application. As the figure indicates, we will use a 2.1 MHz crystal as the time base for the COP452L. With the oscillator divide by 4 selection, this gives an internal frequency period of 1.90476 μs. With this information we can determine the number that needs to be loaded to register A to give a pulse width of 5 ms. From application #1 we have the following equation which is valid here:

$$T = (N + 1)t$$

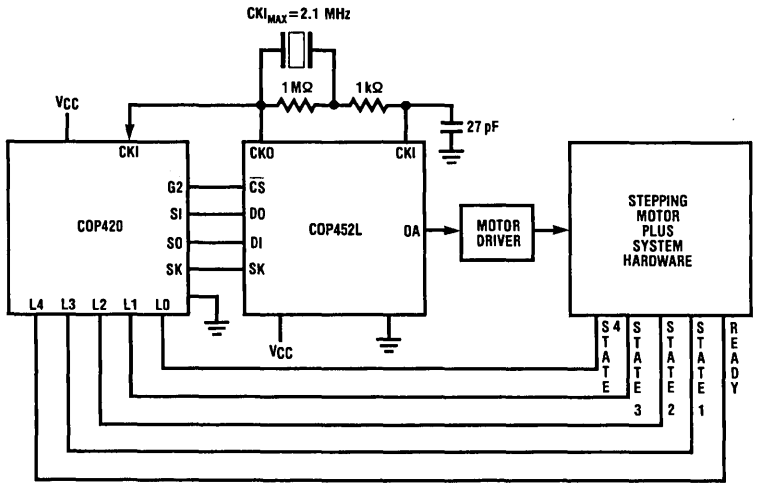
where: T = pulse width
N = contents of register A
t = period of internal clock

Solving for N we have;

$$N = (T/t) - 1$$
$$= (5\ \text{ms}/1.90476\ \mu\text{s}) - 1$$
$$= 2625 - 1$$
$$= 2624$$

Register A must be loaded with 2624 (0A40 hex) to give a 5 ms pulse. The error created by the truncation of the number is 0.5 μs. There is an error of 0.01%—well within the tolerance limits required.

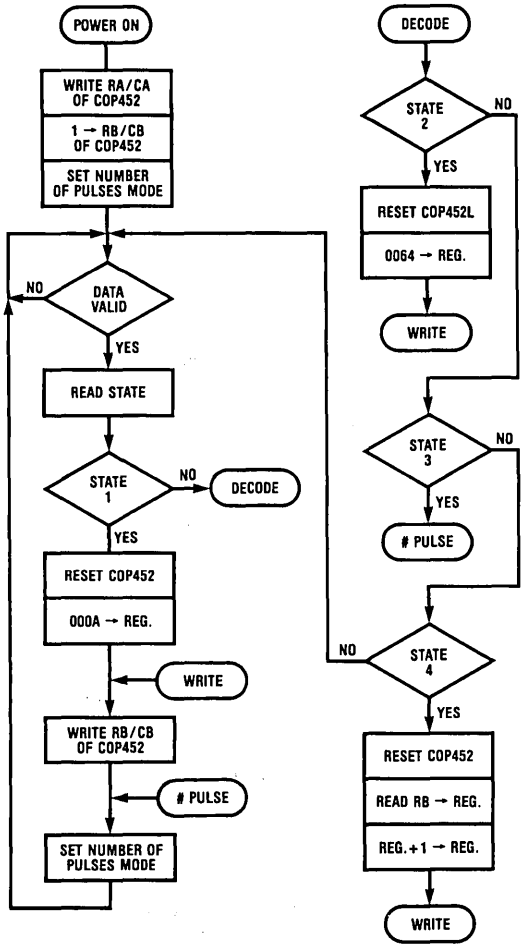
The code to operate this system is given below. The interconnect of *Figure 16* is assumed. The code uses the READ and WRITE subroutines as given in the software interface section of this data sheet. The code further assumes that those routines are located in the subroutine page.



TL/DD/6155–42

FIGURE 16. COP452 In Stepping Motor Control

Applications Information (Continued)



TL/DD/6155–43

FIGURE 17. Flow Diagram for Application #2

```
            . PAGE      0
GSTATE      =           0, 15
POWRON:     CLRA
            XAS                     ; TURN OFF SK CLOCK
            LBI         GSTATE
            STII        15
            LBI         GSTATE
            OMG                     ; DESELECT THE COP452L — G2 HIGH
            LD
            CAMQ                    ; DRIVE THE L LINES HIGH FOR READING
            LEI         4           ; ENABLE THE L OUTPUTS
            LBI         0, 0
            STII        0
            STII        4
            STII        A
            STII        0
            STII        1
            STII        2           ; WRITE RA OF COP452L WITH 0A40 HEX TO GET
            JSRP        WRDATA      ; 5MS PULSE
```

TL/DD/6155–44

Applications Information (Continued)

```
                LBI     0, 0
                STII    5           ; TRANSFER RA TO COUNTER A
                STII    2
                JSRP    WRCMND
                LBI     0, 0        ; NOW WRITE RB WITH THE NUMBER OF PULSES
                STII    1
RBWRT:          STII    0           ; ONE PULSE REQUIRED AT POWER UP
RBWRT2:         STII    0
                STII    0
RBWRT3:         STII    0
                STII    2
                JSRP    WRDATA
                LBI     0, 0        ; NOW TRANSFER RB TO COUNTER B
                STII    4
                STII    2
                JSRP    WRCMND
PULSE:          LBI     0, 0
                STII    2           ; SET NUMBER OF PULSES MODE
                STII    3
                JSRP    WRCMND
; AT THIS POINT THE COP452L IS IN NUMBER OF PULSES MODE, ONE
; PULSE IS OUTPUT AT OA. NOW MUST READ THE RETURN LINES, MAKE
; THE APPROPRIATE DETERMINATION OF THE STATE OF THE SYSTEM
; AND UPDATE THE COP452L ACCORDINGLY, ALSO AT THIE POINT, THE
; COP452L IS SET UP TO AGAIN GENERATE A SINGLE PULSE 5 ms WIDE
; IF THE DEVICE IS ACCESSED AGAIN.
;
STATE:          LBI     GSTATE
                LD                  ; CONTENTS OF GSTATE = 15 HERE
                CAMQ                ; MAKE SURE L LINES ARE HIGH AND
                LEI     4           ; ENABLED
                LBI     0, 0
                INL                 ; READ THE L LINES TO A AND M(0, 0)
                SKMBZ   0           ; TEST DATA — RETURN LINES — VALID
                JMP     STATE       ; DATA NOT VALID, WAIT FOR IT TO BE VALID
                AISC    8           ; DATA IS VALID, DECODE A
                JMP     TEST2
STATE1:         STII    15          ; POINTING AT 0, 0
                STII    3           ; RESET THE COP452L FOR STATE 1
                JSRP    WRCMND
                LBI     0, 0        ; NOW SET UP TO SEND 10 PULSES
                STII    10
                JMP     RBWRIT      ; SHARE COMMON CODE
TEST2:          AISC    4
                JMP     TEST3
STATE2:         STII    15          ; IN STATE2, MUST SEND 100 PULSES
                STII    3           ; FIRST RESET THE COP452L
                JSRP    WRCMND
                LBI     0, 0        ; WRITE 100 (0064 HEX) TO RB OF COP452L
                STII    4
                STII    6
                JMP     RBWRT2
TEST3:          AISC    2
                JMP     TEST4
STATE3:         JMP     PULSE       ; STATE 3 MERELY SENDS THE SAME NUMBER OF PULSES AGAIN.
                                    ; THEREFORE, MERELY SEND THE NUMBER OF PULSES MODE COMMAND
                                    ; AGAIN
```

TL/DD/6155–45

Applications Information (Continued)

```
TEST4:      AISC    1
            JMP     STATE       ; ALL L LINES WERE 0, JUMP BACK TO MAIN
STATE4:     STII    15          ; RESET THE COP452L
            STII    3
            JSRP    WRCMND
            LBI     0, 0        ; NOW READ THE COP452L
            STII    2
            STII    2           ; COMMAND TO READ RB
            JSRP    READ
            LBI     0, 0        ; MOVE DATA TO LAST 4 DIGITS OF R0
            XIS
            XIS
            XIS
            XIS
            LBI     0, 0        ; NOW INCREMENT THE VALUE BY 1
            SC
PLUS1:      CLRA
            ASC
            NOP
            XIS
            CBA
            AISC    12
            JP      PLUS1
            JMP     RBWRT3      ; HAVE INCREMENTED THE VALUE, SEND IT OUT
;
            . PAGE  2
READ:
                                ; SEE SOFTWARE INTERFACE SECTION FOR THESE
WRDATA:                         ; ROUTINES

WRCMND:
```

TL/DD/6155-46

These are general routines and can be reduced in specific applications. The application itself was kept general so that it can be easily adapted to particular applications. The user should view this code as the basis from which to work to optimize the code for a specific application.

APPLICATION #3

An application such as a tachometer requires the counting of external pulses that occur within a given time period. The COP452L can be used both to perform the counting and to establish the "viewing window," or time period, during which to count the pulses. By using the frequency and count mode of the COP452L, a frequency can be generated which will establish this viewing time. The other counter can then be used to count the pulses. *Figure 18* provides a diagram of the interconnect in this application.

As *Figure 18* indicates, the oscillator frequency for the COP452L has been selected as 250 kHz. With the oscillator divider set at divide by 1, the internal frequency is also 250 kHz. At this frequency, the minimum pulse width that can be reliably expected to decrement the counter is 4 μs—the period of the internal frequency.

A viewing time of 250 ms is arbitrarily selected. This means that the period of the output frequency is 500 ms—a frequency of 2 Hz. Using the equation developed earlier for determining the counter values we have:

$$N = \frac{T}{2t} - 1$$

$$= (500\ \text{ms}/8\ \mu\text{s}) - 1$$

$$= 62500 - 1$$

$$N = 62499 = \text{F423 hex}$$

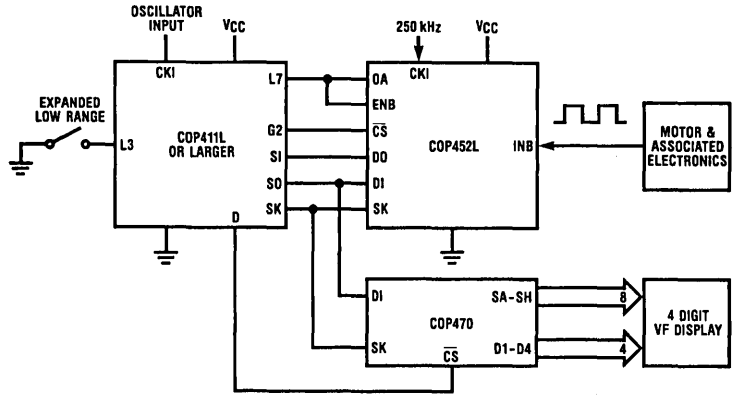
Therefore, register A must be loaded with the hex value F423 to generate a frequency of 2 Hz at OA. Counter B will count pulses when OA is high by virtue of the ENB input. When OA is low, the microcontroller will read and reset the counter and peform any necessary operations.

With the values above for the internal frequency and the viewing window, the tachometer range is 240 RPM to 62,500 RPM. By making use of the divide by 1/divide by 4 features of the oscillator divider, the range can be extended down to 60 RPM. The range when the oscillator is divided by 4 is 60 RPM to 15,625 RPM. However, a penalty is paid for this range extension. The viewing window goes from 250 ms to 1 second. The minimum reliable pulse width also increases from 4 μs to 16 μs. The added time spent counting may or may not be acceptable. It can be reduced somewhat by changing the value of RA to give a faster frequency at the reduced counter clock frequency. However, as the OA frequency increases, the low end of the range increases.

A flow chart for this application is provided in *Figure 19*. Sample code is given below. Note that the sample code includes only the COP452L interface and control. Other system requirements, e.g., display interface, arithmetic, etc., are not included here. Other data sheets and application notes provide sufficient information to fill in those details.

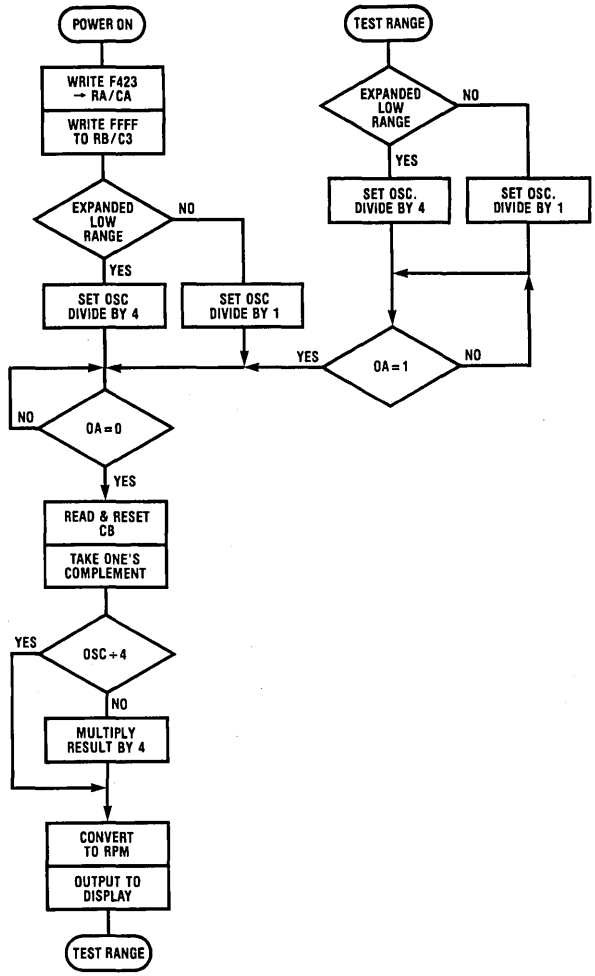
The hardware interface indicated in *Figure 18* and the code below, are completely general and valid of any COPS microcontroller. In specific applications both the hardware and software may be optimized to a greater extent than that shown here.

Applications Information (Continued)



TL/DD/6155-47

FIGURE 18. COP452 In Wide Range Tachometer Application



TL/DD/6155-48

FIGURE 19. Flowchart for Tachometer Application

Applications Information (Continued)

```
                .PAGE   0
GSTATE          =       0, 15
POWRON:         CLRA
                XAS                     ; TURN OFF THE SK CLOCK—C = 0 AT POWER UP
                LBI     GSTATE
                OBD                     ; DRIVE D LINES HIGH TO DESELECT DISPLAY
                STII    15
                LBI     GSTATE
                OMG                     ; DESELECT THE COP452L
                LD
                CAMQ                    ; SET THE Q REGISTER TO ALL 1'S FOR INPUT
                LBI     0, 0
                STII    3               ; NOW SET UP TO WRITE RA OF COP452L
                STII    2
                STII    4
                STII    15              ; WRITE RA WITH F423 HEX
                STII    1                                              TL/DD/6155-49
                STII    2               ; REMEMBER COP452L IS RESET AT POWER UP
                JSRP    WRDATA
                LBI     0, 0
                STII    5               ; TRANSFER RA TO CA
                STII    2
                JSRP    WRCMND
                JSR     RSTRB           ; RESET RB AND COUNTER B WITH FFFF
                JSR     RANGE           ; TEST RANGE AND SET OSCILLATOR DIVIDER
                LEI     4               ; ENABLE Q TO L—DRIVE L LINES HIGH
                LBI     0, 0            ; LOOK FOR OA = 0
TSTOA0:         INL
                SKMBZ   3
                JP      TSTOA0
                LBI     0, 0            ; OA IS 0, READ COUNTER
                STII    6               ; FIRST TRANSFER CB TO RB
                STII    2
                JSRP    WRCMND
                LBI     0, 0            ; THEN READ RB
                STII    2
                STII    2
                JSRP    READ
                LBI     0, 0            ; NOW TAKE THE 1'S COMPLEMENT
ONECMP:         COMP
                XIS
                COMP
                XIS
                COMP
                XIS
                COMP
                X
                LBI     0, 0            ; NOW SAVE VALUE IN R1
XFER1:          LD      1
                XIS     1
                JP      XFER1
                JSR     RSTRB           ; RESET RB AND CB WITH FFFF FOR NEXT TIME
;
; AT THIS POINT INSERT THE APPROPRIATE CODE FOR ANY NECESSARY
; ARITHMETIC, BINARY/BCD CONVERSION, DISPLAY OUTPUT, AND ANY OTHER
; SYSTEM REQUIREMENTS. AFTER THESE ARE COMPLETE, JUMP TO LABEL
; TSTRNG WHICH HAS BEEN ARBITRARILY PLACED IN PAGE 4.
.PAGE           2
WRDATA:

WRCMND:                         ; SEE SOFTWARE INTERFACE SECTION FOR THESE
                                ; THREE ROUTINES
READ:
                .PAGE   4
TSTRNG:         JSR     RANGE   ; CHECK THE RANGE
                LEI     4       ; BE SURE Q IS ENABLED TO L
                LBI     0, 0    ; LOOK FOR OA = 1
TSTOA1:         INL
                SKMBZ   3
                JMP     TSTOA0
                JP      TSTOA1
;
; THE SUBROUTINES RANGE AND RSTRB ARE INSERTED HERE
;
RANGE:          LEI     4       ; MAKE SURE L ENABLED
                LBI     3, 15   ; WILL SAVE RANGE STATUS IN 3, 15
                INL
                X
                CLRA            ; NOW PREPARE TO SET OSCILLATOR DIVIDER        TL/DD/6155-50
```

Applications Information (Continued)

```
                AISC    8           ; AN 8 MEANS DIVIDE BY 1
                SKMBZ   3
                JP      HILOW
LOW:            AISC    1           ; IF DIVIDE BY 4, WANT A 9 IN A
HILOW:          LBI     0, 0
                XIS
                STII    2
                JMP     WRCMND
;
; THE FOLLOWING SUBROUTINE USES A SUBROUTINE LEVEL. IT RESETS BOTH
; REGISTER B AND COUNTER B OF THE COP452L TO FFFF
;
RSTRB:          LBI     0, 0
                STII    15
                STII    15
                STII    15
                STII    15
                STII    0
                STII    2
                JSRP    WRDATA      ; WRITE FFFF TO RB
                LBI     0, 0
                STII    4           ; TRANSFER RB TO CB
                STII    2
                JMP     WRCMND
```

TL/DD/6155–51

APPLICATION #4

The triggered pulse mode of the COP452L provides the capability of generating the appropriate signals for triac control. *Figure 20* is a general diagram of such an application.

Assume the requirement is to switch on the triac 45 degrees into the waveform. With a 60 Hz sine wave signal, the 45 degree delay is 2.0833 ms from the zero crossing. Assume also that the triac requires a gate pulse width of 150 μs. As the diagram indicates, a 2.097 MHz crystal provides the oscillator input to the COP452L. With the above information the two values that must be loaded in the COP452L can be determined. With CKI at 2.097 MHz and the oscillator divider at divide by 4, the period of the internal frequency is 1.9075 μs. From the description of the triggered pulse mode, the pulse width is given by:

$$T = Bt$$

where: T = desired pulse width

B = contents of register B

t = period of internal clock

Solving for B is trivial and gives:

$$B = T/t$$
$$= 150\ \mu s/1.9075\ \mu s$$
$$= 78.64$$

Since the register and counter can be loaded with whole numbers only, register B and counter B must be initialized with 79 (002F hex) to give a pulse width of 150 μs.

The delay from the zero cross trip point is given by:

$$T = (A + 1.5)t$$

where: T = delay from zero cross trip point

A = contents of register A

t = period of internal clock

Solving for A we have:

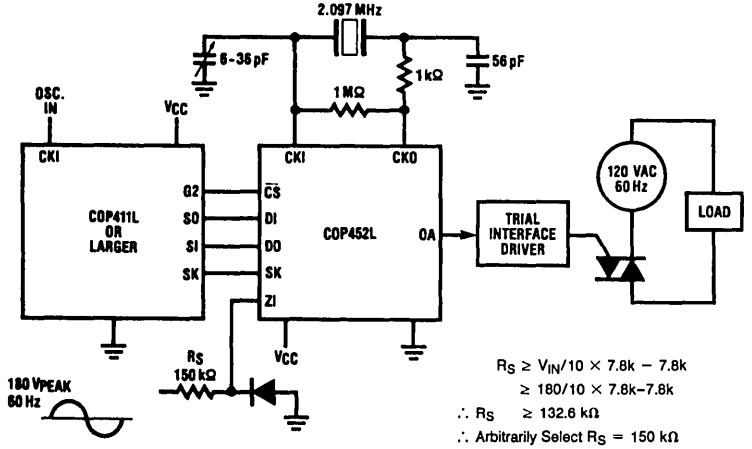
$$A = (T/t) - 1.5$$
$$= (2.0833\ ms/1.9075\ \mu s) - 1.5$$
$$A = 1090.66 \text{ rounded up to } 1091$$

Therefore register A and counter A must be initialized with 1091 (0443 hex) to delay 2.0833 ms (45 degrees at 60 Hz) from zero cross.

Once the data has been given to the COP452L and the device placed in the triggered pulse mode, no further attention is required. The COP452L will generate the pulses with the appropriate delay as long as the power is applied and the input sine wave is available. It is a trivial matter to change any of the information. Merely write the appropriate register/counter pair. Thus very easy control is available over the firing angle of triacs.

Sample code to accomplish this function is given below. The code is general purpose and is written to work in any COPS microcontroller.

Applications Information (Continued)



TL/DD/6155-52

FIGURE 20. COP452L as Triac Controller

```
                .PAGE   0
GSTATE          =       0,15
POWRON:         CLRA
                XAS                     ; TURN OFF THE SK CLOCK
                LBI     GSTATE
                STII    15
                LBI     GSTATE
                OMG                     ; DESELECT THE COP452L — G2 HIGH
                LBI     0,0             ; NOW WRITE RB/CB WITH 002F HEX TO GIVE
                STII    15              ; 150 µs PULSE WIDTH
                STII    2
                STII    0
                STII    0
                STII    0
                STII    2
                JSRP    WRDATA
                LBI     0,0
                STII    4               ; TRANSFER RB TO CB
                STII    2
                JSRP    WRCMND
                LBI     0,0             ; NOW WRITE RA/CA WITH 0443 HEX FOR THE DELAY
                STII    3
                STII    4
                STII    4
                STII    0
                STII    1
                STII    2
                JSRP    WRDATA
                LBI     0,0
                STII    5
                STII    2
                JSRP    WRCMND          ; TRANSFER RA TO CA
                LBI     0,0
                STII    9
```

TL/DD/6155-53

```
                STII    2               ; SET OSCILLATOR DIVIDER TO DIVIDE BY 4
                JSRP    WRCMND
                LBI     0,0
                STII    1               ; SET TRIGGERED PULSE MODE
                STII    3
                JSRP    WRCMND
; ALL COMPLETE AT THIS POINT. ROUTINES WRCMND AND WRDATA ASSUMED
; IN PAGE 2 AND ARE THE SAME AS GIVEN IN SOFTWARE INTERFACE SECTION.
; THE COP452L WILL NOW GENERATE THE 150 µs PULSE DELAYED BY 2.0833 ms
; FROM EVERY ZERO CROSSING. THE USER CAN NOW IGNORE THE TRIAC CONTROL
; AND DO WHATEVER ELSE IS REQUIRED IN THE SYSTEM. FURTHER ATTENTION
; IS REQUIRED ONLY WHEN THE DATA IN THE COP452 MUST BE CHANGED.
```

TL/DD/6155-54

Applications Information (Continued)

Let us now compute the minimum and maximum delays from the true zero crossing in this application. As indicated earlier, the period of the internal frequency here is 1.9075 μs. Counter A contains 0443 hex (decimal 1091). R_S is 150k and the peak input voltage is 180V. A 60 Hz sine wave is assumed. As given earlier, the minimum time is:

$$T_{MIN} = (A+1.5)t - \frac{1}{2\pi f}\arcsin\left(0.15\frac{R_S+2.6k}{V_{IN}\times 2.6k}\right)+0.3\ \mu s$$

Substituting we have:

$$T_{MIN} = 1092.5t - \frac{1}{120\pi}\arcsin\left(0.15\frac{152.6k}{180\times 2.6k}\right)+0.3\ \mu s$$

$$= 2093.9\ \mu s - 129.7\ \mu s + 0.3\ \mu s$$

$$T_{MIN} = 1954.5\ \mu s$$

Similarly, the maximum time is given as:

$$T_{MAX} = (A + 1.5)t + \frac{1}{2\pi f}\arcsin\left(0.15\frac{R_S + 2.6k}{V_{IN}\times 2.6k}\right) + 0.6\ \mu s + \frac{t}{2}$$

Substituting, we have:

$$T_{MAX} = 1092.5t + \frac{1}{120\pi}\arcsin\left(0.15\frac{152.6k}{180\times 2.6k}\right) + 0.6\ \mu s + \frac{1.9075\ \mu s}{2}$$

$$= 2083.9\ \mu s + 129.7\ \mu s + 0.6\ \mu s + 0.9538\ \mu s$$

$$T_{MAX} = 2215.15\ \mu s$$

As is obvious from the preceding analysis, the parameter previously defined as X_1 is the most significant of the additional factors that define the time delay from true zero. This factor can be minimized by using as small a series resistance as possible. The frequency and input voltage will be governed by the application. The user must also remember that the minimum and maximum times calculated in this manner are absolute worst case values derived using the worst case condition.

National Semiconductor

COP470/COP370 V.F. Display Driver

General Description

The COP470 is a peripheral member of National's COPS™ Microcontroller family. It is designed to directly drive a multiplexed Vacuum Fluorescent display. Data is loaded serially and held in internal latches. The COP470 has an on-chip oscillator to multiplex four digits of eight segment display and may be cascaded and/or stacked to drive more digits, more segments, or both.

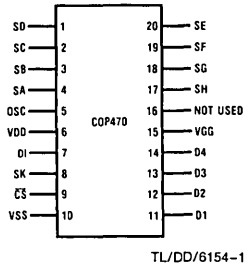
With the addition of external drivers, the COP470 also provides a convenient means of interfacing to a large-digit LED display. The COP370 is the extended temperature range version of the COP470.

Features

- Directly interfaces to multiplexed 4 digit by 8 segment Vacuum Fluorescent displays
- Expandable to drive 8 digits and/or 16 segments
- Compatible with all COP400 processors
- Needs no refresh from processor
- Internal or external oscillator
- No "glitches" on outputs when loading data
- Drives large and small displays
- Programmable display brightness
- Small (20-pin) dual-in-line package
- Operates from 4.5V to 9.5V
- Outputs switch 30V and require no external resistors
- Static latches
- MICROWIRE™ compatible serial I/O
- Extended temperature device COP370 (−40°C to +85°C)

Connection and Block Diagrams

Dual-In-Line Package

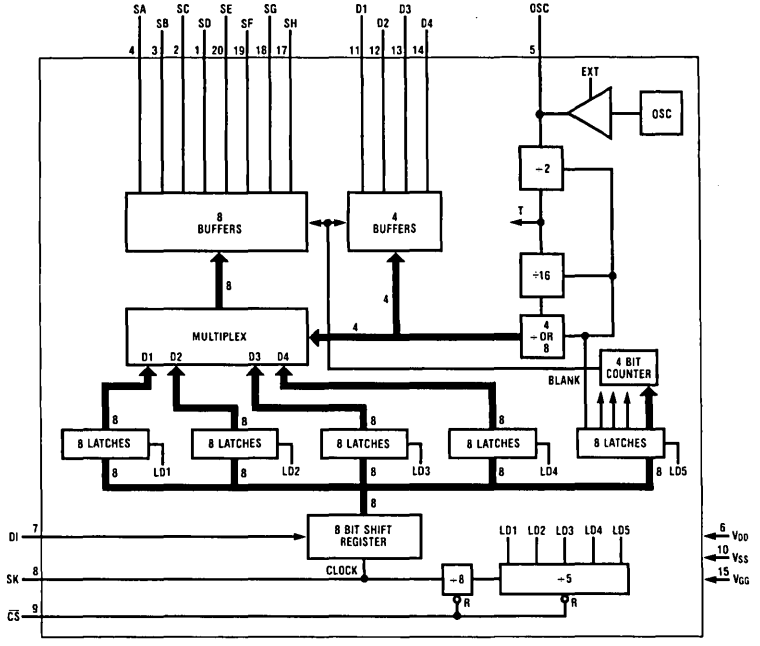


TL/DD/6154-1

Top View

FIGURE 1. COP470

Order Number COP470D, COP370D, COP470N or COP370N
See NS Package Number D20A or N20A



TL/DD/6154-2

FIGURE 2. COP470

Absolute Maximum Ratings (V_{SS} = 0)

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at Display Outputs	+0.3V to −30V
Voltage at All Other Pins	+0.3V to −20V

Operating Temperature	
COP470	0°C to +70°C
COP370	−40°C to +85°C
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Package Power Dissipation	400 mW at 25°C 200 mW at 70°C 125 mW at 85°C

DC Electrical Characteristics

V_{SS} = 0, V_{DD} = −4.5V to −9.5V, V_{GG} = −30V, T_A = 0°C to +70°C for COP470 and T_A = 40°C to 85°C for COP370 unless otherwise specified.

Parameter	Min	Max	Units
Power Supply Voltage			
V_{DD}	−9.5	−4.5	V
V_{GG}	−30	V_{DD}	V
Power Supply Current			
I_{DD}		5	mA
I_{GG} (Displayed Blanked)		1	mA
Input Levels			
V_{IH}	−1.5	+0.3	V
V_{IL}	−10.0	−4.0	V
Output Drive Digits and Segments			
I_{OH} @ V_{OH} = V_{SS} − 3V	10		mA
I_{OH} @ V_{OH} = V_{SS} − 2V	7		mA
I_{OL} @ V_{OL} = V_{GG} + 2V(1)	10		μA
Output Drive @ V_{GG} = V_{DD} = V_{SS} −5V			
I_{OH} @ V_{OH} = V_{SS} −2V	1		mA
Allowable Source Current			
Per Pin		20	mA
Total for Segments		60	mA
Input Capacitance		7	pF
Input Leakage		1	μA

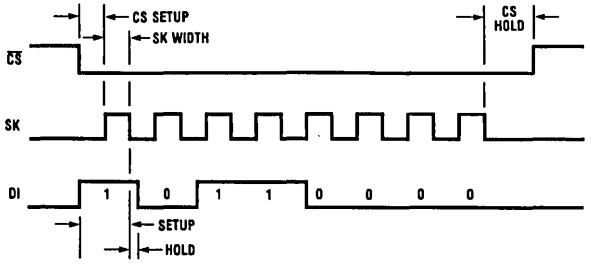
AC Electrical Characteristics

V_{SS} = 0, V_{DD} = −4.5V to −9.5V, V_{GG} = −30V, T_A = 0°C to +70°C for COP470 and T_A = 40°C to 85°C for COP370 unless otherwise specified.

Parameter	Min	Max	Units
OSC Period (internal or external)	4	20	μs
OSC Pulse Width	1.5		μs
Clock Period T (twice Osc. period)	8	40	μs
Display Frequency			
4 digits = 1/64T	390	2000	Hz
8 digits = 1/128T	190	1000	Hz
SK Clock Frequency	0	250	kHz
SK Clock Width	1.5		μs
Data Set-up and Hold Time			
t_{SETUP}	1.0		μs
t_{HOLD}	50		ns
CS Set-up and Hold Time			
t_{SETUP}	1.0		μs
t_{HOLD}	1.0		μs
Duty Cycle			
4 digits	1/64	15/64	
8 digits	1/128	15/128	

Note 1: I_{OL} current is to V_{GG} with the chip running. Current is measured just after the output makes a high-to-low transition.

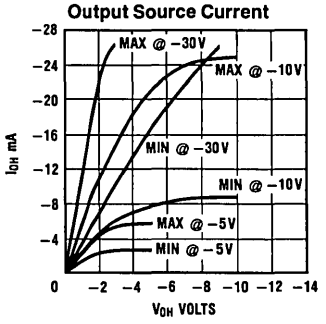
Timing Diagram



TL/DD/6154-3

FIGURE 3. Serial Load Timing Diagram

Performance Characteristic



TL/DD/6154-4

Functional Description

SEGMENT DATA BITS

Data is loaded in serially in sets. Each set of segment data is in the following format:

SA	SB	SC	SD	SE	SF	SG	SH

Data is shifted into an eight bit shift register. The first bit of the data is for segment H, digit 1. The eighth bit is segment A, digit 1.

A set of eight bits is shifted in and then loaded into the digit one latches. The second set of 8 bits is loaded into digit two latches. The third set into digit three latches and the fourth set is loaded into digit four latches.

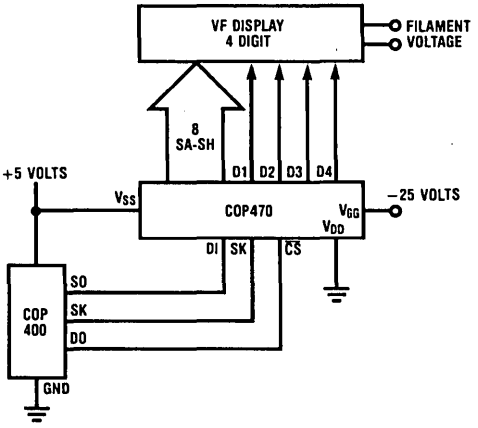
DISPLAY ON TIME AND CONTROL BITS

The fifth set of 8 data bits contains blank time data and control data in the following format:

		Display Digits					
Sync	Ext.	Right	Left	←	On Time		→
	Osc.	4 of 8	4 of 8	LSB			MSB

The first four bits shifted in contain the on time. This is used to control display brightness. The brightness is a function of the on time of each segment divided by the total time (duty cycle). The on time is programmable from 0 to 15 and the total time is 64. For example, if the on time is 15, the duty cycle is 15/64 which is maximum brightness. If on time is 8, the duty cycle is 8/64, about 1/2 brightness. There are 16 levels of brightness from 15/64 to 0/64 (off).

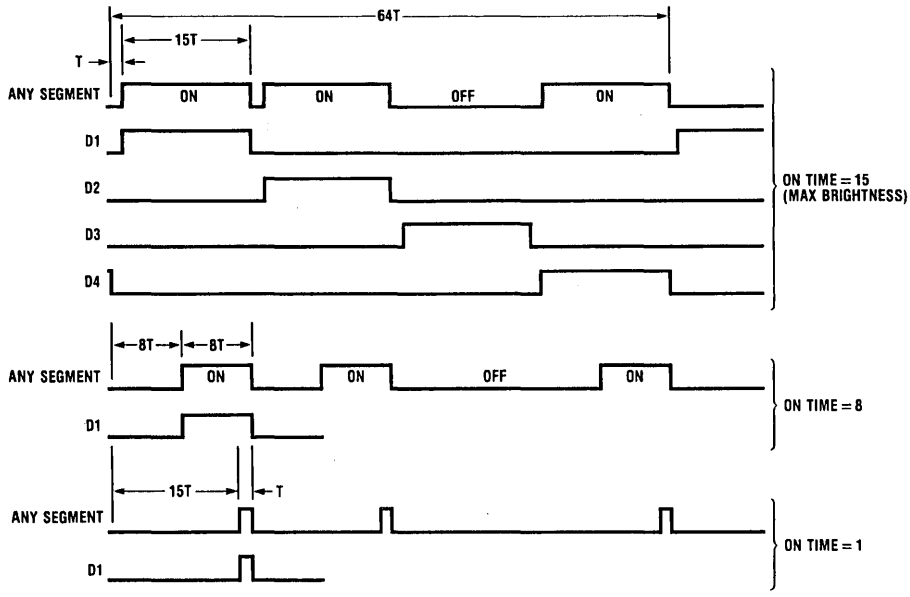
The fifth and sixth bits control the multiplex digits. To enable the COP470 to drive a 4 digit multiplex display, set both bits to one. If two COP470s are used to drive an 8 digit display, bit five is set on the left COP470 and bit six is set on the right COP470 (see *Figure 6*). In the eight digit mode, the display duty cycle is on time/128.



TL/DD/6154-5

FIGURE 4. System Diagram—4 Digit Display

Functional Description (Continued)



TL/DD/6154–6

FIGURE 5. Segment and Digit Output Timing Diagram

The seventh bit selects internal or external oscillator. The OSC pin of the COP470 is either an output of the internal oscillator (bit 7 = 0) or is an input allowing the COP470 to run from an external oscillator (bit 7 = 1).

The eighth bit is set to synchronize two COP470s. For example, to set the COP470 to internal osc, 4 digits, and maximum brightness, send out six ones and two zeros.

LOADING SEQUENCE

Step

1. Turn $\overline{CS}$ Low.
2. Clock in 8 bits of data for digit 1.
3. Clock in 8 bits of data for digit 2.
4. Clock in 8 bits of data for digit 3.
5. Clock in 8 bits of data for digit 4.
6. Clock in 8 bits of data for on time and control bits.
7. Turn $\overline{CS}$ high.

Note: $\overline{CS}$ may be turned high after any step. For example, to load only 2 digits of data do steps 1, 2, 3, and 7. $\overline{CS}$ must make a high to low transition before loading data in order to reset internal counters.

8 DIGIT Displays

Two COP470s may be tied together in order to drive an eight digit multiplexed display. This is shown in *Figure 6.* The following is the loading sequence to drive an eight digit display using two COP470s.

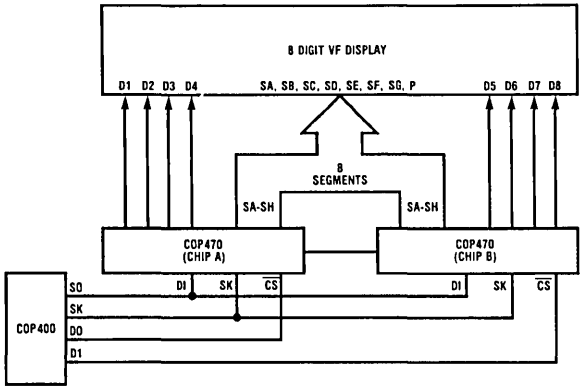
1. Turn $\overline{CS}$ lwo on both COP470s.
2. Shift in 32 bits of data for the right 4 digits.
3. Shift in 4 bits of on time, a zero and three ones. This synchronizes both chips, sets to external oscillator, and to right four of eight digits. Thus both chips are synchronized and the oscillator is stopped.
4. Turn $\overline{CS}$ high to both chips.
5. Turn $\overline{CS}$ low to the left COP470.
6. Shift in 32 bits of data for the left 4 digits.
7. Shift in 4 bits of on time, a one and three zeros. This sets this COP470 to internal oscillator and to left four of eight digits. Now both chips start and run off the same oscillator.
8. Turn $\overline{CS}$ high.

The chips are now synchronized and driving eight digits of display. To load new data simply load each chip separately in the normal manner.

16 SEGMENT DISPLAY

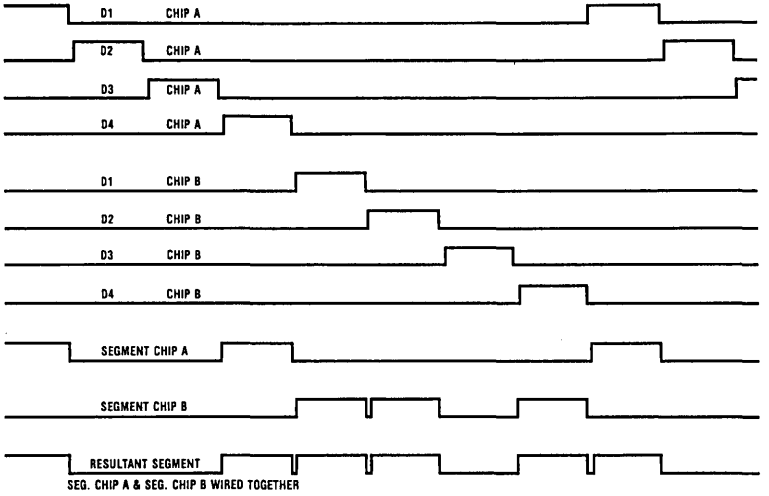
Two COP470s may be tied together in order to drive a sixteen segment display. This is shown in *Figure 8.* To do this, both chips must be synchronized, one must run off external oscillator while the other runs off its internal oscillator outputting to the other. Similarly, four COP470s could be tied together to drive eight digits of sixteen segments.

Functional Description (Continued)



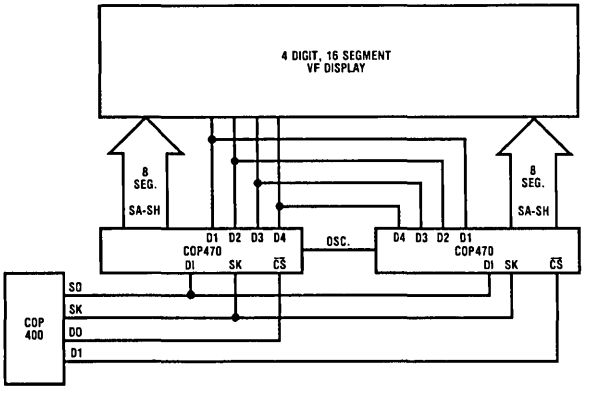
TL/DD/6154–7

FIGURE 6. System Diagram 8 Digit Display



TL/DD/6154–8

FIGURE 7. Segment and Digit Output Timing Diagram for 8 Digits



TL/DD/6154–9

FIGURE 8. System Diagram for 16 Segment Display

Functional Description (Continued)

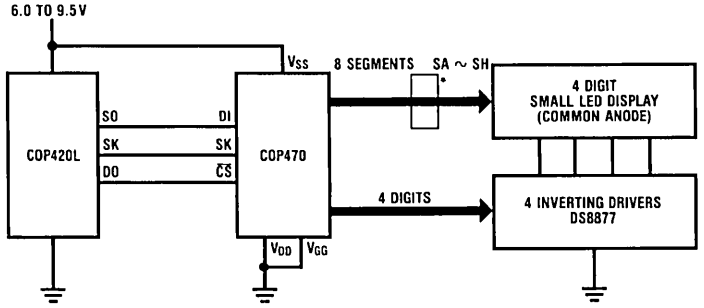
LED DISPLAY

The COP470 may be used to drive LED displays. The COP470 can drive the segments directly on small, low current LED displays as shown in *Figure 9.* By adding display drivers, large, high current LED displays can be driven as shown in *Figure 10.*

Example:
COP420 Code to Load COP470
(Display Data is in Memory 0, 12–0, 15)

```
        LBI 0,12        ; Point to first display data
        OBD             ; Turn CS low (DO)
LOOP:   CLRA
        LQID            ; Look up segment data
        CQMA            ; Copy data from Q to M & A
        SC              ; Set C to turn on SK
        XAS             ; Output lower 4 bits of data
        NOP             ; Delay
        NOP             ; Delay
        LD              ; Load A with upper 4 bits
        XAS             ; Output 4 bits of data
        NOP             ; Delay
        NOP             ; Delay
        RC              ; Reset C
        XAS             ; Turn off SK clock
        XIS             ; Increment B for next data
        JP LOOP         ; Skip this jump after last digit
        SC              ; Set C
        CLRA            ;
        AISC 15         ; 15 to A
        XAS             ; Output on time (max brightness)
        NOP             ;
        CLRA            ;
        AISC 12         ; 12 to A
        XAS             ; Output control bits
        NOP             ;
        LBI 0,15        ; 15 to B
        RC              ; Reset C
        XAS             ; Turn off SK
        OBD             ; Turn CS high (DO)
```

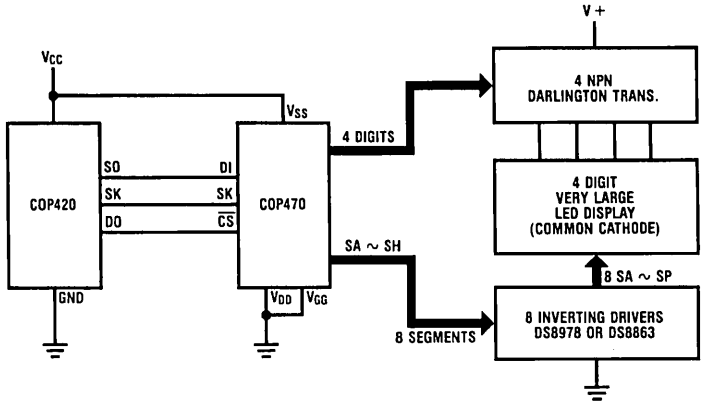
Functional Description (Continued)



TL/DD/6154–10

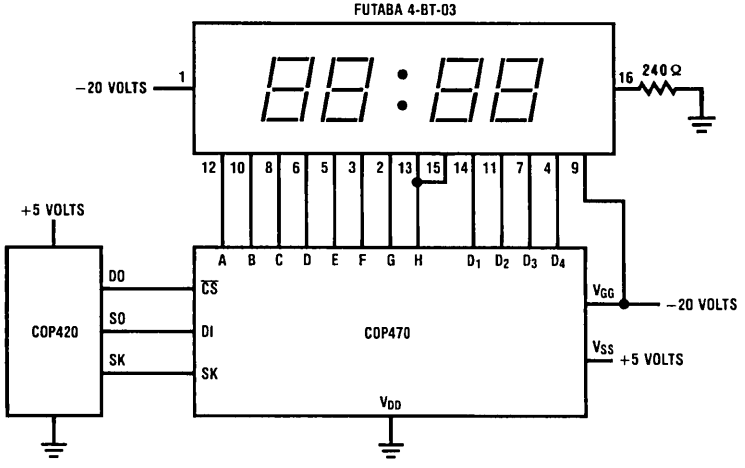
*Segment buffer may be added for larger display.

FIGURE 9. LED Display



TL/DD/6154–11

FIGURE 10. Large LED Display



TL/DD/6154–12

FIGURE 11. Sample V.F. System

National Semiconductor

COP472-3 Liquid Crystal Display Controller

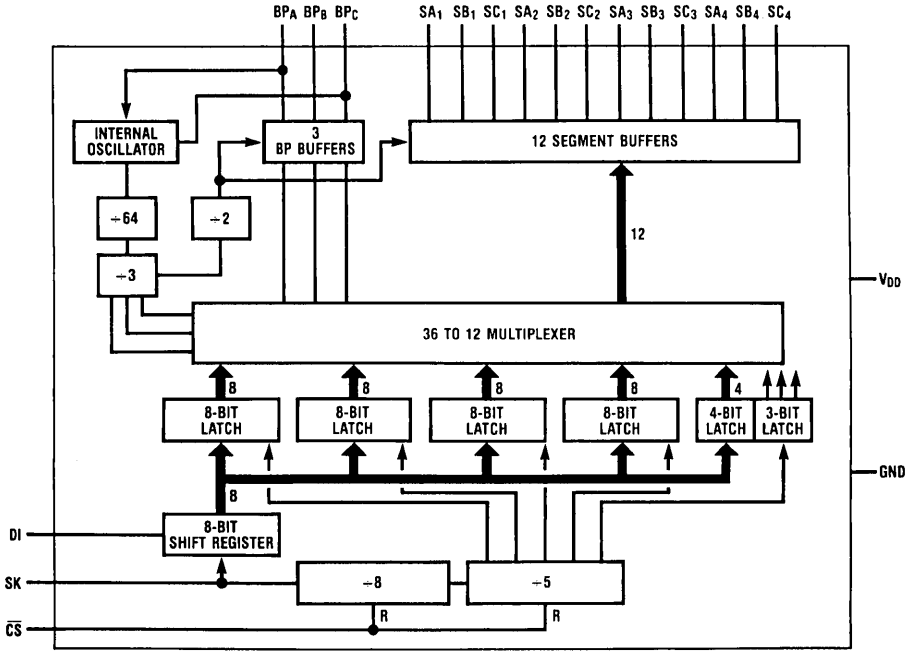
General Description

The COP472-3 Liquid Crystal Display (LCD) Controller is a peripheral member of the COPS™ family, fabricated using CMOS technology. The COP472-3 drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472-3 contains an on-chip oscillator and generates all the multi-level waveforms for backplanes and segment outputs on a triplex display. One COP472-3 can drive 36 segments multiplexed as 3 x 12 (4½ digit display). Two COP472-3 devices can be used together to drive 72 segments (3 x 24) which could be an 8½ digit display.

Features

- Direct interface to TRIPLEX LCD
- Low power dissipation (100 μW typ.)
- Low cost
- Compatible with all COP400 processors
- Needs no refresh from processor
- On-chip oscillator and latches
- Expandable to longer displays
- Software compatible with COP470 V.F. Display Driver chip
- Operates from display voltage
- MICROWIRE™ compatible serial I/O
- 20-pin Dual-In-Line package

Block Diagram



TL/DD/6932-1

Absolute Maximum Ratings

Voltage at CS, DI, SK pins	−0.3V to +9.5V	Storage Temperature	−65°C to +150°C
Voltage at all other Pins	−0.3V to V_{DD}+0.3V	Lead Temp. (Soldering, 10 Seconds)	300°C
Operating Temperature Range	0°C to 70°C		

DC Electrical Characteristics

GND = 0V, V_{DD} = 3.0V to 5.5V, T_A = 0°C to 70°C (depends on display characteristics)

Parameter	Conditions	Min	Max	Units
Power Supply Voltage, V_{DD}		3.0	5.5	Volts
Power Supply Current, I_{DD} (Note 1)	V_{DD}=5.5V		250	μA
	V_{DD}=3V		100	μA
Input Levels				
DI, SK, CS				
V_{IL}			0.8	Volts
V_{IH}		0.7 V_{DD}	9.5	Volts
BPA (as Osc. in)				
V_{IL}			0.6	Volts
V_{IH}		V_{DD}−0.6	V_{DD}	Volts
Output Levels, BPC (as Osc. Out)				
V_{OL}			0.4	Volts
V_{OH}		V_{DD}−0.4	V_{DD}	Volts
Backplane Outputs (BPA, BPB, BPC)				
$V_{BPA, BPB, BPC}$ ON	During	$V_{DD}-\Delta V$	V_{DD}	Volts
$V_{BPA, BPB, BPC}$ OFF	BP+ Time	$\frac{1}{3} V_{DD}-\Delta V$	$\frac{1}{3} V_{DD}+\Delta V$	Volts
$V_{BPA, BPB, BPC}$ ON	During	0	ΔV	Volts
$V_{BPA, BPB, BPC}$ OFF	BP− Time	$\frac{2}{3} V_{DD}-\Delta V$	$\frac{2}{3} V_{DD}+\Delta V$	Volts
Segment Outputs (SA_1 ~ SA_4)				
V_{SEG} ON	During	0	ΔV	Volts
V_{SEG} OFF	BP+ Time	$\frac{2}{3} V_{DD}-\Delta V$	$\frac{2}{3} V_{DD}+\Delta V$	Volts
V_{SEG} ON	During	$V_{DD}-\Delta V$	V_{DD}	Volts
V_{SEG} OFF	BP− Time	$\frac{1}{3} V_{DD}-\Delta V$	$\frac{1}{3} V_{DD}+\Delta V$	Volts
Internal Oscillator Frequency		15	80	kHz
Frame Time (Int. Osc. ÷ 192)		2.4	12.8	ms
Scan Frequency ($1/T_{SCAN}$)		39	208	Hz
SK Clock Frequency		4	250	kHz
SK Width		1.7		μs
DI				
Data Setup, t_{SETUP}		1.0		μs
Data Hold, t_{HOLD}		100		ns
$\overline{CS}$				
t_{SETUP}		1.0		μs
t_{HOLD}		1.0		μs
Output Loading Capacitance			100	pF

Note 1: Power supply current is measured in stand-alone mode with all outputs open and all inputs at V_{DD}.

Note 2: $\Delta V = 0.05 V_{DD}$.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Voltage at CS, DI, SK Pins	−0.3V to +9.5V
Voltage at All Other Pins	−0.3V to V_{DD}+0.3V
Operating Temperature Range	−40°C to +85°C
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

DC Electrical Characteristics

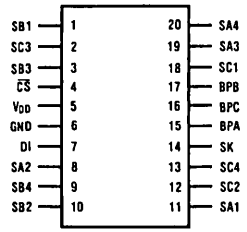
GND = 0V, V_{DD} = 3.0V to 5.5V, T_A = −40°C to +85°C (depends on display characteristics)

Parameter	Conditions	Min	Max	Units
Power Supply Voltage, V_{DD}		3.0	5.5	Volts
Power Supply Current, I_{DD} (Note 1)	V_{DD}=5.5V		300	μA
	V_{DD}=3V		120	μA
Input Levels				
DI, SK, CS				
V_{IL}			0.8	Volts
V_{IH}		0.7 V_{DD}	9.5	Volts
BPA (as Osc. In)				
V_{IL}			0.6	Volts
V_{IH}		V_{DD}−0.6	V_{DD}	Volts
Output Levels, BPC (as Osc. Out)				
V_{OL}			0.4	Volts
V_{OH}		V_{DD}−0.4	V_{DD}	Volts
Backplane Outputs (BPA, BPB, BPC)				
$V_{BPA, BPB, BPC}$ ON	During	V_{DD}−ΔV	V_{DD}	Volts
$V_{BPA, BPB, BPC}$ OFF	BP+ Time	⅓ V_{DD}−ΔV	⅓ V_{DD}+ΔV	Volts
$V_{BPA, BPB, BPC}$ ON	During	0	ΔV	Volts
$V_{BPA, BPB, BPC}$ OFF	BP− Time	⅔ V_{DD}−ΔV	⅔ V_{DD}+ΔV	Volts
Segment Outputs (SA_1 ~ SA_4)				
V_{SEG} ON	During	0	ΔV	Volts
V_{SEG} OFF	BP+ Time	⅔ V_{DD}−ΔV	⅔ V_{DD}+ΔV	Volts
V_{SEG} ON	During	V_{DD}−ΔV	V_{DD}	Volts
V_{SEG} OFF	BP− Time	⅓ V_{DD}−ΔV	⅓ V_{DD}+ΔV	Volts
Internal Oscillator Frequency		15	80	kHz
Frame Time (Int. Osc. ÷ 192)		2.4	12.8	ms
Scan Frequency ($1/T_{SCAN}$)		39	208	Hz
SK Clock Frequency		4	250	kHz
SK Width		1.7		μs
DI				
Data Setup, t_{SETUP}		1.0		μs
Data Hold, t_{HOLD}		100		ns
$\overline{CS}$				
t_{SETUP}		1.0		μs
t_{HOLD}		1.0		μs
Output Loading Capacitance			100	pF

Note 1: Power supply current is measured in stand-alone mode with all outputs open and all inputs at V_{DD}.

Note 2: ΔV = 0.05 V_{DD}.

Dual-In-Line Package



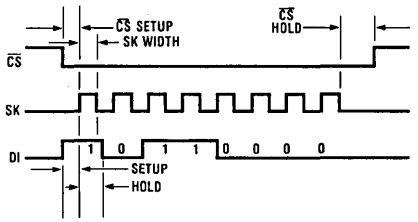
Top View

TL/DD/6932-2

Pin	Description
$\overline{CS}$	Chip select
V_{DD}	Power supply (display voltage)
GND	Ground
DI	Serial data input
SK	Serial clock input
BP_A	Display backplane A (or oscillator in)
BP_B	Display backplane B
BP_C	Display backplane C (or oscillator out)
SA1 ~ SC4	12 multiplexed outputs

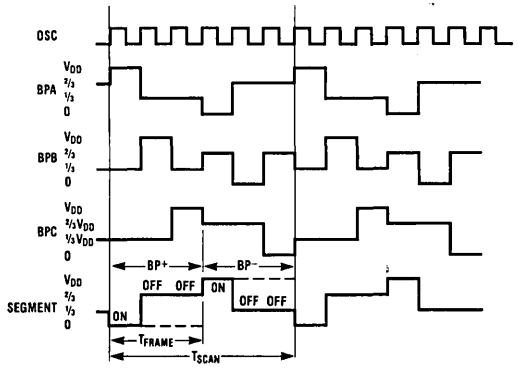
Order Number COP472MW-3 or COP472N-3
See NS Package Number M20A or N20A

FIGURE 2. Connection Diagram



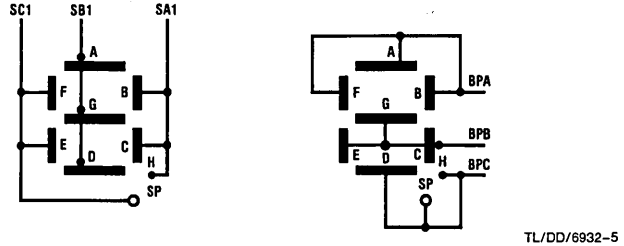
TL/DD/6932-3

FIGURE 3. Serial Load Timing Diagram



TL/DD/6932-4

FIGURE 4. Backplane and Segment Waveforms



TL/DD/6932-5

FIGURE 5. Typical Display Internal Connections
Epson LD-370

Functional Description

The COP472-3 drives 36 bits of display information organized as twelve segments and three backplanes. The COP472-3 requires 40 information bits: 36 data and 4 control. The function of each control bit is described below. Display information format is a function of the LCD interconnections. A typical segment/backplane configuration is illustrated in *Figure 5*, with this configuration the COP472-3 will drive 4 digits of 9 segments.

To adapt the COP472-3 to any LCD display configuration, the segment/backplane multiplex scheme is illustrated in Table I.

Two or more COP472-3 chips can be cascaded to drive additional segments. There is no limit to the number of COP472-3's that can be used as long as the output loading capacitance does not exceed specification.

TABLE I. COP472-3 Segment/Backplane Multiplex Scheme

Bit Number	Segment, Backplane	Data to Numeric Display
1	SA1, BPC	SH
2	SB1, BPB	SG
3	SC1, BPA	SF
4	SC1, BPB	SE
5	SB1, BPC	SD
6	SA1, BPB	SC
7	SA1, BPA	SB
8	SB1, BPA	SA
9	SA2, BPC	SH
10	SB2, BPB	SG
11	SC2, BPA	SF
12	SC2, BPB	SE
13	SB2, BPC	SD
14	SA2, BPB	SC
15	SA2, BPA	SB
16	SB2, BPA	SA
17	SA3, BPC	SH
18	SB3, BPB	SG
19	SC3, BPA	SF
20	SC3, BPB	SE
21	SB3, BPC	SD
22	SA3, BPB	SC
23	SA3, BPA	SB
24	SB3, BPA	SA
25	SA4, BPC	SH
26	SB4, BPB	SG
27	SC4, BPA	SF
28	SC4, BPB	SE
29	SB4, BPC	SD
30	SA4, BPB	SC
31	SA4, BPA	SB
32	SB4, BPA	SA
33	SC1, BPC	SPA
34	SC2, BPC	SP2
35	SC3, BPC	SP3
36	SC4, BPC	SP4
37	not used	
38	Q6	
39	Q7	
40	SYNC	

Digit 1 (bits 1–8); Digit 2 (bits 9–16); Digit 3 (bits 17–24); Digit 4 (bits 25–32); Digit 1 (bit 33); Digit 2 (bit 34); Digit 3 (bit 35); Digit 4 (bit 36)

SEGMENT DATA BITS

Data is loaded in serially, in sets of eight bits. Each set of segment data is in the following format:

SA	SB	SC	SD	SE	SF	SG	SH

Data is shifted into an eight bit shift register. The first bit of the data is for segment H, digit 1. The eighth bit is segment A, digit 1. A set of eight bits is shifted in and then loaded into the digit one latches. The second set of 8 bits is loaded into digit two latches. The third set into digit three latches, and the fourth set is loaded into digit four latches.

CONTROL BITS

The fifth set of 8 data bits contains special segment data and control data in the following format:

SYNC	Q7	Q6	X	SP4	SP3	SP2	SP1

The first four bits shifted in contain the special character segment data. The fifth bit is not used. The sixth and seventh bits program the COP472-3 as a stand alone LCD driver or as a master or slave for cascading COP472-3's. BPC of the master is connected to BPA of each slave. The following table summarizes the function of bits six and seven:

Q7	Q6	Function	BPC Output	BPA Output
1	1	Slave	Backplane Output	Oscillator Input
0	1	Stand Alone	Backplane Output	Backplane Output
1	0	Not Used	Internal Osc. Output	Oscillator Input
0	0	Master	Internal Osc. Output	Backplane Output

The eighth bit is used to synchronize two COP472-3's to drive an 8½-digit display.

LOADING SEQUENCE TO DRIVE A 4½-DIGIT DISPLAY

Steps:

1. Turn $\overline{CE}$ low.
2. Clock in 8 bits of data for digit 1.
3. Clock in 8 bits of data for digit 2.
4. Clock in 8 bits of data for digit 3.
5. Clock in 8 bits of data for digit 4.
6. Clock in 8 bits of data for special segment and control function of BPC and BPA.

0	0	1	1	SP4	SP3	SP2	SP1

7. Turn $\overline{CS}$ high.

Note: $\overline{CS}$ may be turned high after any step. For example to load only 2 digits of data, do steps 1, 2, 3, and 7.

$\overline{CS}$ must make a high to low transition before loading data in order to reset internal counters.

LOADING SEQUENCE TO DRIVE AN 8½-DIGIT DISPLAY

Two or more COP472-3's may be connected together to drive additional segments. An eight digit multiplexed display is shown in *Figure 7*. The following is the loading sequence to drive an eight digit display using two COP472-3's. The right chip is the master and the left the slave.

Steps:

1. Turn $\overline{CS}$ low on both COP472-3's.
2. Shift in 32 bits of data for the slave's four digits.
3. Shift in 4 bits of special segment data: a zero and three ones.

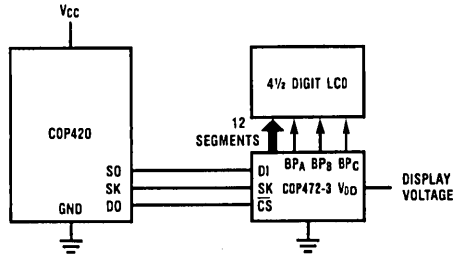
1	1	1	0	SP4	SP3	SP2	SP1

   This synchronizes both the chips and BPA is oscillator input. Both chips are now stopped.
4. Turn CS high to both chips.
5. Turn CS low to master COP472-3.
6. Shift in 32 bits of data for the master's 4 digits.
7. Shift in four bits of special segment data, a one and three zeros.

0	0	0	1	SP4	SP3	SP2	SP1

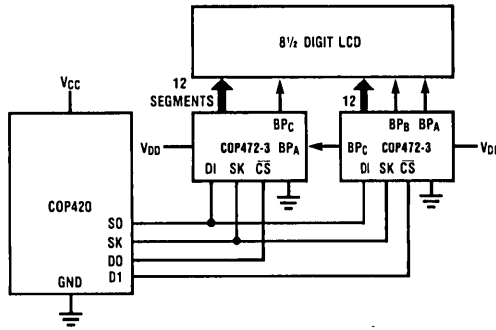
   This sets the master COP472-3 to BPA as a normal backplane output and BPC as oscillator output. Now both the chips start and run off the same oscillator.
8. Turn $\overline{CS}$ high.

The chips are now synchronized and driving 8 digits of display. To load new data simply load each chip separately in the normal manner, keeping the correct status bits to each COP472-3 (0110 or 0001).



TL/DD/6932-6

FIGURE 6. System Diagram – 4½ Digit Display



TL/DD/6932-7

FIGURE 7. System Diagram – 8½ Digit Display

Example Software

Example 1

COP420 Code to load a COP472-3 [Display data is in M(0, 12)-M(0, 15), special segment data is in M(0, 0)]

```
                LBI 0, 12          ; POINT TO FIRST DISPLAY DATA
                OBD                ; TURN CS LOW (DO)
LOOP:           CLRA
                LQID               ; LOOK UP SEGMENT DATA
                CQMA               ; COPY DATA FROM Q TO M & A
                SC                 ; SET C TO TURN ON SK
                XAS                ; OUTPUT LOWER 4 BITS OF DATA
                NOP                ; DELAY
                NOP                ; DELAY
                LD                 ; LOAD A WITH UPPER 4 BITS
                XAS                ; OUTPUT 4 BITS OF DATA
                NOP                ; DELAY
                NOP                ; DELAY
                RC                 ; RESET C
                XAS                ; TURN OFF SK CLOCK
                XIS                ; INCREMENT B FOR NEXT DATA
                JP LOOP            ; SKIP THIS JUMP AFTER LAST DIGIT
                SC                 ; SET C
                LBI 0, 0           ; ADDRESS SPECIAL SEGMENTS
                LD                 ; LOAD INTO A
                XAS                ; OUTPUT SPECIAL SEGMENTS
                NOP                ;
                CLRA               ;
                AISC 12            ; 12 to A
                XAS                ; OUTPUT CONTROL BITS
                NOP                ;
                LBI 0, 15          ; 15 to B
                RC                 ; RESET C
                XAS                ; TURN OFF SK
                OBD                ; TURN CS HIGH (DO)
```

Example Software (Continued)

Example 2

COP420 Code to load two COP472-3 parts [Display data is in M(0, 12)-M(0, 15) and M(1, 12)-M(1, 15), special segment data is in M(0, 0) and M(1, 0)]

```
INIT:               LBI     0, 15
                    OBD                     ; TURN BOTH CS'S HIGH
                    LEI     8               ; ENABLE SO OUT OF S. R.
                    RC
                    XAS                     ; TURN OFF SK CLOCK
                    LBI     3, 15           ; USE M(3, 15) FOR CONTROL BITS
                    STII    7               ; STORE 7 TO SYNC BOTH CHIPS
                    LBI     0, 12           ; SET B TO TURN BOTH CS'S LOW
                    JSR     OUT             ; CALL OUTPUT SUBROUTINE
MAIN DISPLAY SEQUENCE
DISPLAY             LBI     3, 15
                    STII    8               ; SET CONTROL BITS FOR SLAVE
                    LBI     0, 13           ; SET B TO TURN SLAVE CS LOW
                    JSR     OUT             ; OUTPUT DATA FROM REG. 0
                    LBI     3, 15
                    STII    6               ; SET CONTROL BITS FOR MASTER
                    LBI     1, 14           ; SET B TO TURN MASTER CS LOW
                    JSR     OUT             ; OUTPUT DATA FROM REG. 1
OUTPUT SUBROUTINE
OUT:                OBD                     ; OUTPUT B TO CS'S
                    CLRA
                    AISC    12              ; 12 TO A
                    CAB                     ; POINT TO DISPLAY DIGIT (BD=12)
LOOP                CLRA
                    LQID                    ; LOOK UP SEGMENT DATA
                    CQMA                    : COPY DATA FROM Q TO M & A
                    SC
                    XAS                     ; OUTPUT LOWER 4 BITS OF DATA
                    NOP                     ; DELAY
                    NOP                     ; DELAY
                    LD                      ; LOAD A WITH UPPER 4 BITS
                    XAS                     ; OUTPUT 4 BITS OF DATA
                    NOP                     ; DELAY
                    NOP                     ; DELAY
                    RC                      ; RESET C
                    XAS                     ; TURN OFF SK
                    XIS                     ; INCREMENT B FOR NEXT DISPLAY DIGIT
                    JP      LOOP            ; SKIP THIS JUMP AFTER LAST DIGIT
                    SC                      ; SET C
                    NOP
                    LD                      ; LOAD SPECIAL SEGS. TO A (BD=0)
                    XAS                     ; OUTPUT SPECIAL SEGMENTS
                    NOP
                    LBI     3, 15
                    LD                      ; LOAD A
                    XAS                     ; OUTPUT CONTROL BITS
                    NOP
                    NOP
                    RC
                    XAS                     ; TURN OFF SK
                    OBD                     ; TURN CS'S HIGH (BD=15)
                    RET
```

National Semiconductor

COP498/COP398 Low Power CMOS RAM and Timer (RAT™) COP499/COP399 Low Power CMOS Memory

General Description

The COP498/398 Low Power CMOS RAM and Timer (RAT) and the COP499/399 Memory are peripheral members of the COPS™ family, fabricated using low power CMOS technology. These devices provide external data storage and/or timing, and are accessed via the simple MICROWIRE™ serial interface. Each device contains 256 bits of read/write memory organized into 4 registers of 64 bits each; each register can be serially loaded or read by a COPS controller.

The COP498/398 also contain a crystal-based timer for timekeeping purposes, and can provide a "wake-up" signal to turn on a COPS controller. Hence, these devices are ideal for applications requiring very low power drain in a standby mode, while maintaining a real-time clock (e.g., electronically-tuned automobile radio). Power is minimized by cycling controller power off for periods of time when no processing is required.

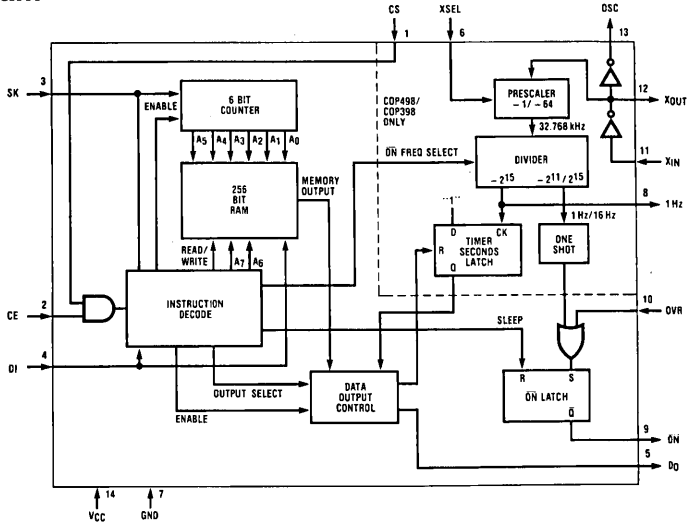
The COP499/399 contain circuitry that enables the user to turn a controller on and off while maintaining the integrity of the memory.

A COP400 series N-channel microcontroller coupled with a COP498 (or 499) RAM/Timer offers a user the low-power advantages of an all CMOS system and the low-cost advantage of an NMOS system. This type of system is ideally suited to a wide variety of automotive and instrumentation applications.

Features

- Low power dissipation
- Quiescent current = 40 nA typical (25°C, V_{CC} = 3.0V)
- Low cost
- Single supply operation (2.4V–5.5V)
- CMOS-compatible I/O
- 4 x 64 serial read/write memory
- Crystal-based selectable timer—2.097152 MHz or 32.768 kHz (COP498/398)
- Software selectable 1 Hz or 16 Hz "wake-up" signal for COPS controller (COP498/398)
- External override to "wake-up" controller
- Compatible with all COP400 processors (processor $V_{CC} \leq 9.5V$)
- MICROWIRE-compatible serial I/O
- Memory protection with write enable and write disable instructions
- 14-pin Dual-In-Line package (COP498/398) or 8-pin Dual-In-Line package (COP499/399)

Block Diagram



TL/DD/6684–1

FIGURE 1

Absolute Maximum Ratings

Voltage relative to GND	
At XSEL, 1 Hz, X_{IN}, X_{OUT}, DO	−0.3V to V_{CC} +0.3V
At all other pins	−0.3V to 10V
Maximum V_{CC} Voltage	6.5V
Total Sink Current Allowed	15 mA
Total Source Current Allowed	10 mA
Ambient Operating Temperature	
COP398/COP399	−40°C to +85°C
COP498/COP499	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation	50 mW

Note: *"Absolute maximum ratings" indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not insured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

COP398/COP399: −40°C ≤ T_A ≤ +85°C unless otherwise specified.

COP498/COP499: 0°C ≤ T_A ≤ +70°C unless other wise specified.

Parameter	Conditions	Min	Max	Units
Operating Voltage	COP498/COP499	2.4	5.5	V
	COP398/COP399	3.0	5.5	V
Quiescent Current	All inputs at GND			
	T_A = 25°C, V_{CC} = 3.0V		1.0	µA
	T_A = 25°C, V_{CC} = 5.0V		3.0	µA
	T_A = 25°C, V_{CC} = 5.5V		6.0	µA
	T_A = 70°C, V_{CC} = 3.0V		4.0	µA
	T_A = 70°C, V_{CC} = 5.0V		10	µA
	T_A = 70°C, V_{CC} = 5.5V		20	µA
(COP398/COP399 only)	T_A = 85°C, V_{CC} = 3.0V		8.0	µA
	T_A = 85°C, V_{CC} = 5.0V		16	µA
	T_A = 85°C, V_{CC} = 5.5V		30	µA
COP498/COP398				
Standby Current (sleep mode)	V_{CC} = Min., Osc. = 2.097 MHz		200	µA
(running with crystal	V_{CC} = Max., Osc. = 2.097 MHz		700	µA
	V_{CC} = Min., Osc. = 32.768 kHz		20	µA
	V_{CC} = Max., Osc. = 32.768 kHz		100	µA
Operating Current	SK = 250 kHz square wave			
	V_{CC} = Min., Osc. = 2.097 MHz		300	µA
	V_{CC} = Max., Osc. = 2.097 MHz		920	µA
	V_{CC} = Min., Osc. = 32.768 kHz		120	µA
	V_{CC} = Max., Osc. = 32.768 kHz		320	µA
COP499/COP399 Operating Current	SK = 250 kHz square wave			
	V_{CC} = 2.4V for COP498/COP499		100	µA
	V_{CC} = 3.0V for COP398/COP399		140	µA
	V_{CC} = Max.		250	µA
Input Voltage Levels				
CE Input	(Schmitt Trigger Input)			
Logic High (V_{IH})		0.8V_{CC}		V
Logic Low (V_{IL})			0.4V_{CC}	V
OVR Input	(Schmitt Trigger Input)			
Logic High (V_{IH})		0.8V_{CC}		V
Logic Low (V_{IL})			0.2V_{CC}	V
All Other Inputs				
Logic High (V_{IH})		0.7V_{CC}		V
Logic Low (V_{IL})			0.3V_{CC}	V
Output Voltage Levels—DO, 1 Hz				
CMOS Operation				
Logic High (V_{OH})	I_{OH} = −10 µA	V_{CC}−0.1		V
Logic Low (V_{OL})	I_{OL} = 10 µA		0.1	V

DC Electrical Characteristics (Continued)

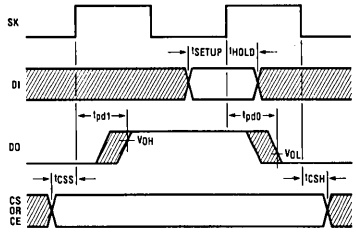
Parameter	Conditions	Min	Max	Units
Input Leakage Current	COP498/COP499, $V_{IH} = V_{CC}$, $V_{IL} = 0V$	−1.0	+1.0	μA
	COP398/COP399, $V_{IH} = V_{CC}$, $V_{IL} = 0V$	−2.0	+2.0	μA
TRI-STATE®, Open Drain	COP498/COP499, $V_H = V_{CC}$, $V_L = 0V$	−2.5	+2.5	μA
Leakage Current	COP398/COP399, $V_H = V_{CC}$, $V_L = 0V$	−5.0	+5.0	μA
Output Current Levels	$V_{CC} = 4.5V$			
Sink Current				
OSC	$V_{OL} = 0.4V$	0.5		mA
$\overline{ON}$	$V_{OL} = 1.5V$	1.5	8.5	mA
X_{OUT}	XSEL = 1, $X_{IN} = 4.5V$, $V_{OL} = 1.0V$	0.25		mA
X_{OUT}	XSEL = 0, $X_{IN} = 4.5V$, $V_{OL} = 2.0V$	8.0		μA
1 Hz, DO	$V_{OL} = 0.8V$	0.8		mA
Source Current				
$\overline{ON}$	$V_{OH} = 1.0V$	60		μA
X_{OUT}	XSEL = 1, $X_{IN} = 0V$, $V_{OH} = 3.0V$	0.27		mA
X_{OUT}	XSEL = 0, $X_{IN} = 0V$, $V_{OH} = 3.0V$	10		μA
1 Hz, DO	$V_{OH} = 2.0V$	0.4		mA

AC Electrical Characteristics

COP398/COP399: $-40°C \le T_A \le +85°C$ unless otherwise specified.

COP498/COP499: $0°C \le T_A \le +70°C$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
COP Interface				
SK Frequency	CS = 1, CE = 1 COP498/COP499	4.096	250	kHz
	CS = 1, CE = 1 COP398/COP399	8.192	250	kHz
SK Duty Cycle	SK frequency ≥ 25 kHz	25	75	%
	SK frequency = 4.096 kHz	48	52	%
Inputs				
CS				
t_{CSS}		0.2		μs
t_{CSH}		0		μs
DI				
t_{SETUP}		0.4		μs
t_{HOLD}		0.4		μs
Output				
DO	$C_L = 100$ pF, $4.5V \le V_{CC} \le 5.5V$,			
t_{pd1}, t_{pd0}	$V_{OUT} = 1.5V$		2.0	μs
t_{pd1}, t_{pd0}	$C_L = 50$ pF, V_{CC} = Min.,			
	$V_{OUT} = 1.5V$		2.4	μs
Crystal Osc. Frequency	XSEL = 1		2.1	MHz
	XSEL = 0		65	kHz

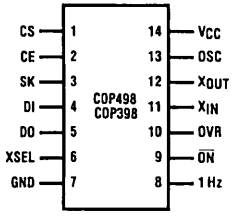


TL/DD/6684–2

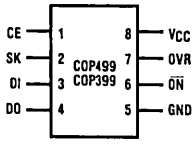
FIGURE 2. Synchronous Data Timing

Connection Diagrams

Dual-In-Line Package



Top View



TL/DD/6684-3

FIGURE 3

Order Number COP398N, COP498N, COP399N, or COP499N
See NS Package Number N08E or N14A

Pin Descriptions

Pin	Description
CS	Chip Select
CE	Chip Enable
SK	Serial Data Clock
DI	Serial Data Input
DO	Serial Data Output
XSEL	Crystal Option Select
X_{IN}	Crystal Oscillator Input
X_{OUT}	Crystal Oscillator Output

Pin	Description
1 Hz	1 Hz Square Wave Output
$\overline{ON}$	Active Low Wake-Up Signal to COPS Controller
OVR	External Override Wake-Up for COPS Controller
OSC	Open Drain Oscillator Output
V_{CC}	Power Supply
GND	Ground

COP398 and COP399 are extended temperature devices (−40°C to +85°C) of COP498 and COP499 (0°C to 70°C) respectively, with all other functional and electrical characteristics being the same. Therefore, no further attempt will be made to distinguish between COP498 and COP398 or between COP499 and COP399. Unless otherwise specified, the following descriptions will apply to both COP498 and COP499, and they will be known as the device.

INSTRUCTION SET

COP498 has six instructions as indicated in *Figure 4.* Note that the MSB of any given instruction is a "1". This bit is properly viewed as a start bit in the interface sequence. The lower 4 bits of the instruction contain the command for the device. One of the instructions (TSEC) should not be used in COP499 as it serves no purpose.

Instruction	Opcode (MSB)	Comments
WRITE	1 s 1 r_1 r_0	s = $\overline{ON}$ (wake up signal) frequency select 1 = 16 Hz, 0 = 1 Hz (s selection for COP498 only) (s = 0 for COP499)
READ	1 1 0 r_1 r_0	r_1, r_0 = register number (00, 01, 10, 11)
WREN	1 0 0 1 1	Write enable
WRDS	1 0 0 0 0	Write disable
TSEC	1 0 0 1 0	Test timer seconds latch (COP498 only)
SLEEP	1 0 0 0 1	Put COPS controller to sleep ($\overline{ON}$ high)

FIGURE 4. Instruction Set

Functional Description

A block diagram of COP498 and COP499 is given in *Figure 1.* Positive logic is used. When a bit is set to the higher voltage it is a logic "1"; when a bit is reset to the lower voltage it is a logic "0". The COP498 can execute six instructions: READ (from any one of 4 registers in memory); WRITE (to any one of 4 registers in memory); WREN (write enable); WRDS (write disable); TSEC (test and reset timer seconds latch); and SLEEP (drive $\overline{ON}$ signal high to turn off COPS controller). The COP499 can execute all the above instructions except TSEC. All communications with the device are via the serial MICROWIRE interface. Both CS and CE (CE only in COP499) must be high to enable the device. The device must be deselected between instructions — either CS and/or CE must go low to insure proper operation. The deselecting of the device resets the counters and serial input register.

READ/WRITE MEMORY

The device has 256 bits of read/write memory. The memory is organized as 4 registers of 64 bits each. The data is accessed serially through the Data input (DI) and Data Output (DO) pins. SK is the clock signal for data and instructions.

The memory address register can be conceived of as two registers: one two bits long and loaded directly from the instruction; the other six bits long and incremented by 1 with each SK pulse as long as the chip is selected. The two bit register does not change during the execution of a given instruction. The six bit register is reset to zero while the device is deselected. When counting, the six bit register wraps around from its maximum value back to zero. Thus memory locations are addressed relative to the number of SK pulses after the chip is selected.

Functional Description (Continued)

The READ instruction will select one of the 4 registers (the register being identified in the instruction opcode as indicated in *Figure 4*) and output the contents of that register to the DO pin until the device is deselected. Note that data output from the device, as a result of a READ instruction, continues as long as the device is selected and clocks are provided. Reading more than 64 bits will cause rereading of some bits as the memory address register wraps around from the maximum value back to zero.

The WRITE instruction selects one of the 4 registers (the register being identified in the instruction opcode as indicated in *Figure 4*) and takes the data from the DI pin and stores that data into the memory register until the device is deselected. The write Operation continues as long as the device is selected and clocks are provided. Thus writing more than 64 bits will cause a portion of the data to be overwritten.

TIMER (COP498 ONLY)

With the XSEL pin tied high (V_{CC}), the timer is a 21 stage counter which can divide a 2.097152 MHz signal down to 1 Hz. This creates the 1 Hz signal output. With XSEL tied low (ground), the timer is a 15 stage counter which divides a 32.768 kHz signal down to create the 1 Hz signal output. The rising edge of the 1 Hz signal is used internally to set the timer seconds latch. A wake-up signal is generated at the $\overline{ON}$ output. This signal can be used to turn a COPS controller on. The wake-up rate is software selectable and may be either 1 Hz or 16 Hz. A bit in the WRITE instruction controls this wake-up rate (see *Figure 4*). By means of the SLEEP instruction a COPS controller may cause the $\overline{ON}$ signal to go high thereby providing a means for the controller to safely turn itself off.

An override capability is present whereby the $\overline{ON}$ pin may be prevented from going high. A "1" level at the OVR pin will force $\overline{ON}$ to go low (or stay low) thereby causing the controller to turn on or remain on. $\overline{ON}$ will remain low, and the controller on, as long as the OVR pin is high. To preserve timekeeping when using the override feature, a timer seconds latch is provided. This latch is set by the rising edge of the 1 Hz signal and is read and reset by the TSEC instruction. The timer seconds latch is primarily intended for use when the override feature is implemented. However, it does provide a convenient one second timer which is software testable over a common serial port.

SYSTEM CONSIDERATIONS

When the COPS processor is being turned on and off, during the power supply transition between ground and operating voltage, some pulses may occur at the output pins of the processor. By using the WRDS and WREN instructions, together with the higher "1" level of the CE pin, accidental writing into the memory may be prevented. This is done by disabling the write operation before going to sleep and enabling the write operation when the COPS processor starts execution. A WRDS instruction is automatically executed if the SLEEP instruction causes $\overline{ON}$ to go high turning off the COPS processor. Furthermore, WREN instruction is disabled as long as $\overline{ON}$ remains high.

The XSEL pin, which identifies the timer counter length, should be tied to either V_{CC} or ground depending on the crystal input. For proper operation, the state of XSEL should not be changed while the device is in operation. If the oscillator and timer features are not used, the X_{IN} pin should be connected to the GND pin and XSEL tied to V_{CC}. If the override feature is not used the OVR pin should be connected to the GND pin.

The device is in a static mode when either the CS or CE pin is low. However, the device is in a dynamic mode when both CS and CE are high and at least one high level has been detected at SK while both pins are high. Because of this, a minimum frequency is specified for the SK clock. This minimum frequency really translates to maximum on and off times for the SK clock. As the SK clock slows down, the duty cycle must get closer to 50%. For best operation, the user should regard the maximum on and off times for the SK clock as about 122 μs (61 μs for COP398/COP399).

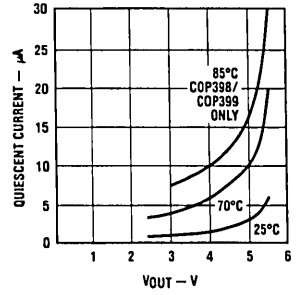
COPS CONTROLLER TO COP498/COP499 HARDWARE INTERFACE

If the COPS controller is operating with a 4 μs instruction cycle time, a 47k resistor should be connected between SK and V_{CC} to speed up the rise time of the SK clock. If the override feature is used in COP498, the override signal should be connected to the OVR pin of the COP498 and an input of the COPS controller. This is simply to provide a means for the controller to know if it was turned on by override or normal timeout. The override signal should be free of noise. In systems where the COPS controller is operating with V_{CC} greater than 6 volts, SI and the override input on the controller should have high impedance, standard TTL level input options selected. To minimize current drain in the controller, the override input to the controller should always use the high impedance option.

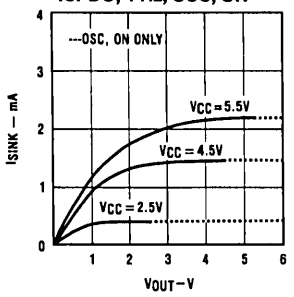
Figure 6a illustrates the COP498 interface in a system with supply voltage less than 6 volts. The COPS controller can either be turned on by the timer or an external signal. A PNP transistor, controlled by the $\overline{ON}$ signal of the COP498, is used to gate the power to the COPS controller. A 0.05 μF capacitor is connected across the supply pins of the controller to reduce voltage variations due to current spikes. It is not recommended to use large capacitance values here as problems can be introduced if the power supply fall time is too long. The switched supply fall time should be kept to about ten instruction cycles of the COPS processor. Resistor R2, between the $\overline{ON}$ pin of the COP498 and the base of the transistor, is used to limit current. Resistor R1, between the base and emitter of the transistor, is used to turn the transistor off when $\overline{ON}$ is high. The CE pin of the COP498 is tied to the V_{CC} pin of the controller. This guarantees that the controller is at its full operating voltage before the COP498 can be accessed. When turned on, the PNP transistor should be saturated in order to minimize the voltage drop across it. The system power supply, which here is V_{CC} to the COP498, must be high enough to insure that the controller V_{CC} — which is the system supply less the voltage drop across the PNP transistor — is high enough to be recognized as a logic "1" at the CE input of the COP498. It is also desirable to have all input signals to the COP498 as close as possible to the COP498 supply levels to eliminate any static power drain which could significantly increase standby and operating current.

Typical Performance Curves

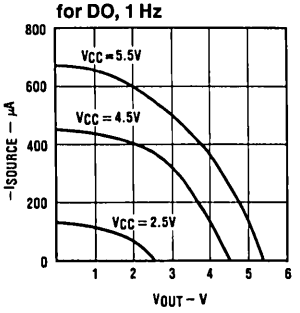
Maximum Quiescent Current COP498/499/398/399



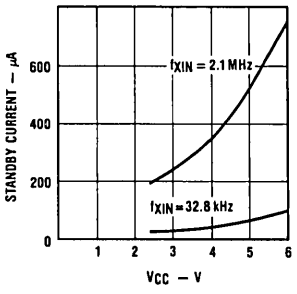
Minimum Sink Current for DO, 1 Hz, OSC, $\overline{ON}$



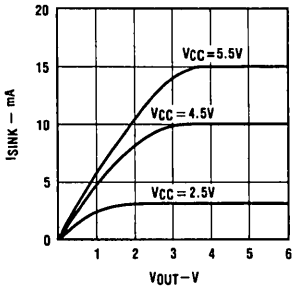
Minimum Source Current for DO, 1 Hz



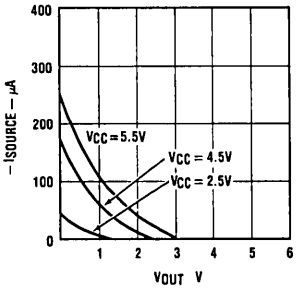
Maximum Standby Current for COP498/398



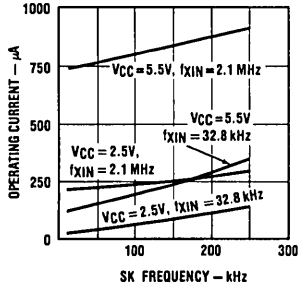
Maximum Sink Current for $\overline{ON}$



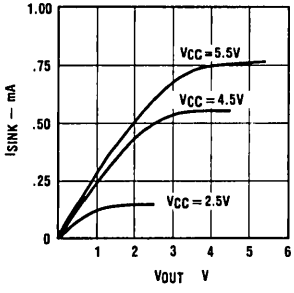
Minimum Source Current for $\overline{ON}$



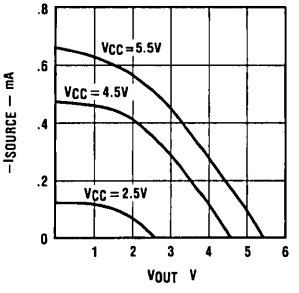
Maximum COP498/398 Operating Current



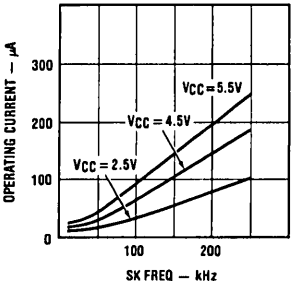
X_{OUT} Minimum Sink Current with XSEL = 1



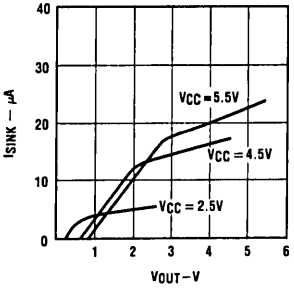
X_{OUT} Minimum Source Current with XSEL = 1



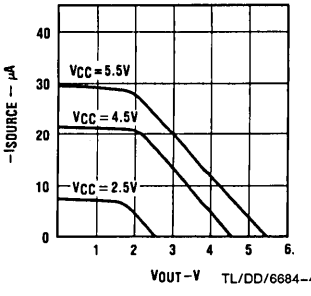
Maximum COP499/399 Operating Current



X_{OUT} Minimum Sink Current with XSEL = 0



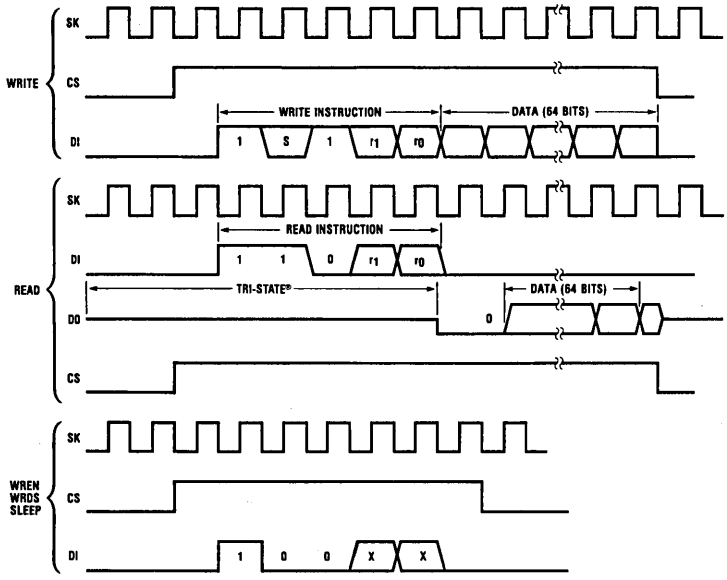
X_{OUT} Minimum Source Current with XSEL = 0



TL/DD/6684–4

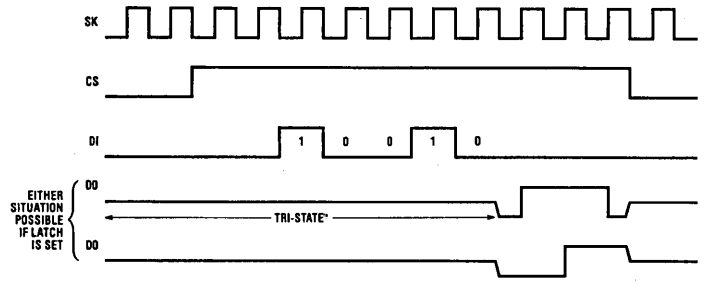
6

Functional Description (Continued)



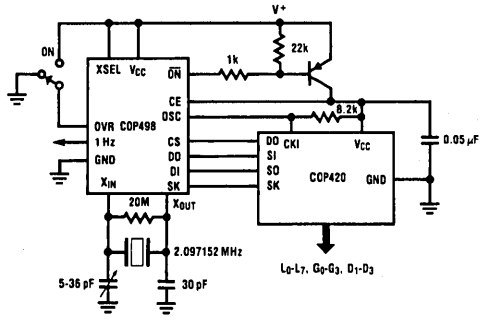
TL/DD/6684-5

FIGURE 5a. Instruction Timing



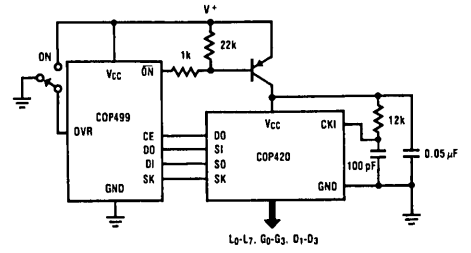
TL/DD/6684-6

FIGURE 5b. TSEC Instruction Timing



TL/DD/6684-7

FIGURE 6a. COP498-COP420 Interface



TL/DD/6684-8

FIGURE 6b. COP499-COP420 Interface

Functional Description (Continued)

Figure 6b illustrates the COP499 interface in a system with a supply voltage less than 6 volts. The COPS processor is being turned on by a switch (or an external signal) connected to the OVR pin.

Figure 7 illustrates a COP498 interface in a system with a supply voltage greater than 6 volts. In such a system, the COP498 cannot be connected directly across the system supply. The power to the COP498 is derived from the system supply by means of a standard zener diode arrangement. A zener diode with a breakdown of about 5 volts is recommended. A capacitor is connected across the COP498 supply pins to reduce voltage variations due to current spikes and to supply extra current when the COP498 is in active operation. Here it is assumed that the COP498 is in standby mode, i.e., deselected, most of the time and is active, selected, for a short period (less than 100 SK periods).

The zener diode series resistor R3 should be selected to meet the current requirements of the zener diode and the standby current of the device. The primary purpose of the zener diode is to place an upper limit on the value of V_{CC} to the device. This insures that V_{CC} to the device will not exceed the specified maximum value. Since the device will operate from 2.5V to 6.0V, the choice of zener diode and series resistor is not critical.

Note that the user may generate the two supply voltages in any manner compatible with system requirements.

Because the COPS controller and the device have different operating voltages, the high impedance standard TTL level input should be selected on the COPS controller for SI and any other input to the controller from the device.

SAMPLE SYSTEM CURRENT DRAIN CALCULATION

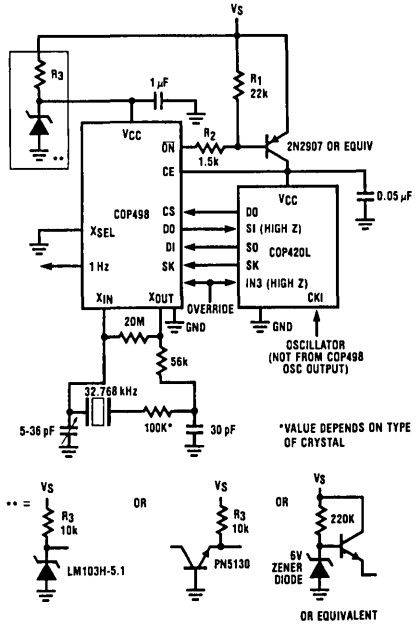
Suppose a 5V system consists of a COP420 and a COP498 with a 32.768 kHz crystal. The COP420 is being turned on once a second. Assume that the COP420 need 10 ms for internal reset and 10 ms to update all the necessary information, then the COP420 will be turned on for 20 ms every second, i.e., a duty cycle of 2%; and the COP498 will be in operating mode for at most 10 ms, i.e., a duty cycle of less than 1%. Because of the short duty cycle, it is further assumed that the COP498 current drain will be that of standby current, about 75 μA at 5V. The current drain through the base of the switching transistor that turns on the COP420 can be estimated by the voltage drop across the current limiting resistor and in this case is assumed to be 3.5 mA.

COP498 current drain = 75 μA
COP420 current drain = 0.02 × 25 mA = 500 μA
Switching transistor base current = 0.02 × 3.5 mA = 70 μA
Total system current drain = 500 + 70 + 75 μA = 645 μA

The result shows that it is possible to achieve the low cost of NMOS and low power dissipation of CMOS simultaneously with a system consisting of a COP498 and a COPS processor.

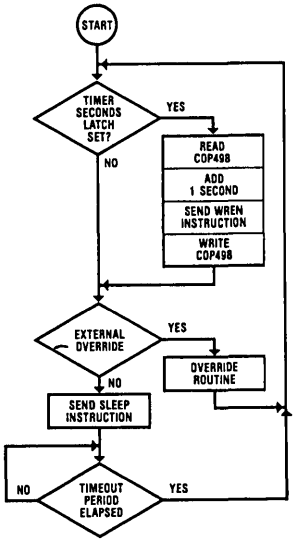
COPS CONTROLLER — COP498/398 SOFTWARE INTERFACE

Figure 8 shows a typical flow chart for a COP498 or COP499 interface to a COPS microcontroller system. This flow chart also illustrates the override feature. Since the override feature is being used, the first step is to inquire the device if it is necessary to increment the time. It is assumed that timekeeping is a necessary part of the application. This interrogation of the device is accomplished by means of the TSEC instruction which dumps the contents of the timer seconds latch to the serial output port and resets the latch. If the latch was set, the time must be incremented. This is accomplished by reading the appropriate memory register into the controller, incrementing the time and writing the register back out to the device. The next step is to check for the override signal. If it is present a special override routine may be performed. If no override is present, the controller



TL/DD/6684-9

FIGURE 7. COP498-COP420L Interface with V_S = 9V and 32.768 kHz Crystal



TL/DD/6684-10

FIGURE 8. Typical COP498 Interface Flowchart

Functional Description (Continued)

turns itself off by sending a SLEEP command to the device. After sending the SLEEP command, the controller goes into a loop to wait for power to go off. In the event the controller is turned back on by the override signal before the voltage has dropped, the loop has a time limit which, when exceeded, causes the controller to jump to the beginning of the program and start again. If the override feature is not used there is no need to test the timer seconds latch nor to test for the override signal. Without the override, the controller can only be turned on by the COP498 if the time out period has elapsed. Note also that the timer features continue to operate regardless of the state of the override signal. The override signal, when high, merely forces the $\overline{ON}$ pin to go low. The operation of the rest of the chip is in no way affected by the override signal.

GENERAL CODE FOR SOFTWARE INTERFACE

The code in *Figure 9a* is recommended for interfacing the device to any COPS controller other than COP410L/COP411L. The code in *Figure 9b* is the recommended interface code for COP410L/COP411L. The code is written as subroutines and the code uses one level of subroutine internally. It is apparent from the code that the software interface is somewhat different for the READ and WRITE instructions than for the rest of the instructions. The routine labelled SETUP is assumed to be in page 2 of the ROM. The rest of the code may be located anywhere in program memory subject to the usual programming rules of COPS microcontrollers. The lower four bits of the instruction opcode are assumed to be located in RAM location COMAND, which is chosen as location 3,15. Data I/O uses register 2. The controller-COP498/499 interface is assumed to be as in *Figure 6* or *Figure 7*. It is assumed that the SIO register in the COPS controller is enabled as serial I/O prior to entry to these routines.

```
WRITE:  JSRP   SETUP
RW:     LD
        XAS             ; READ/WRITE DATA
        XIS
        JP     RW
        OBD              ; DISABLE THE COP498/499 (B=0)
        JP     FINISH
READ:   JSRP   SETUP
        NOP              ; NEED A TOTAL OF 5 SK CLOCK DELAYS (5 NOP'S)
        NOP              ; UNTIL DATA OUT IS VALID AT SIO REGISTER
        NOP
        NOP
        NOP
        JP     RW
INSTRT: JSRP   SETUP     ; ROUTINE FOR THE REST OF THE INSTRUCTIONS
        NOP
        NOP              ; DELAYS TO INSURE PROPER TIMING
FINISH: CLRA
        RC
        OBD              ; DESELECT THE COP498/499 (B=0)
        XAS              ; TURN OFF THE CLOCK
        RET
        . PAGE 2
SETUP:  LBI    COMMAND   ; POINT TO LOCATION WHERE COMMAND STORED
        CLRA
        SC
        XAS              ; TURN ON SK CLOCK
        OBD              ; ENABLE THE COP498/499 (B=15)
        CLRA
        XAS              ; MAKE SURE NO INVALID DATA SENT
        CLRA
        AISC   1         ; SET UP START BIT
        SC
        XAS              ; SEND START BIT MSD OF INSTRUCTION
        LD               ; FETCH COMMAND TO A
        NOP
        NOP              ; MAINTAIN PROPER TIMING
        XAS              ; SEND COMMAND
        LBI    2,0       ; POINT TO READ/WRITE REGISTER
        RET              ; RETURN TO MAIN ROUTINE
```

FIGURE 9a. Software Interface to COP498/COP499 for COPS Controllers Other Than COP410L/COP411L

Functional Description (Continued)

```
WRITE:   JSRP   SETUP
RW1:     XAS                ; SEND COMMAND
RW2:     LD
         XDS                ; POSITION Bd PROPERLY
RW:      LD
         XAS
         XIS
         JP     RW
         OBD                ; DISABLE THE COP498/499 (B=0)
         JP     FINISH
READ:    JSRP   SETUP
         XAS                ; SEND READ COMMAND
         NOP                ; DELAY FOR DATA VALID
         NOP
         NOP
         NOP
         NOP
         JP     RW2
INSTRT:  JSRP   SETUP       ; ROUTINE FOR REST OF INSTRUCTIONS
         XAS                ; SEND INSTRUCTION
         NOP
         NOP
         NOP                ; DELAY FOR INSTRUCTION ACCEPT
         NOP
FINISH:  CLRA
         RC
         OBD                ; DESELECT THE COP498/499
         XAS                ; TURN OFF THE CLOCK
         RET
         . PAGE 2
SETUP:   LBI    COMMAND
         CLRA
         SC
         XAS                ; TURN ON SK CLOCK
         OBD                 ENABLE THE COP498/COP499 (B=15)
         CLRA
         XAS                ; MAKE SURE NO INVALID DATA SENT
         CLRA
         AISC   1
         SC
         XAS                ; SEND START BIT-MSD OF INSTRUCTION
         LD                 ; FETCH INSTRUCTION
         LBI    2,9
         RET
```

FIGURE 9b. COP410L/COP411L Software Interface to COP498/COP499

The code in *Figure 9a* will read or write 64 bits at a time. Note that in the COP410L/411L the code in *Figure 9b* will read or write 32 bits at a time. The code of *Figure 10* is recommended if the user wishes to work in blocks of 64 bits with the COP410L/411L. Only the code which is different from that shown in *Figure 9b* is shown in *Figure 10.*

The routine in *Figure 10* will read/write into registers 2 and 1 in the COP410L/411L. *Figure 10* illustrates the preferred method of achieving full utilization of the device memory when the COP410L/411L is the contoller. Remember that all the other routines are as shown in *Figure 9B. Figure 10* illustrates only that code that must be changed to achieve full usage of the device memory when using the COP410L/411L.

GENERAL NOTES

1. For complete safety in all cases it is recommended that the SK clock be turned off after the device has been deselected since the device is dynamic when it is enabled, If the clock is turned off while the device is selected, special care must be given to the SK timing characteristics. In no case should the clock be turned off while the device is selected if the SK period is greater than about 50 μs.

Functional Description (Continued)

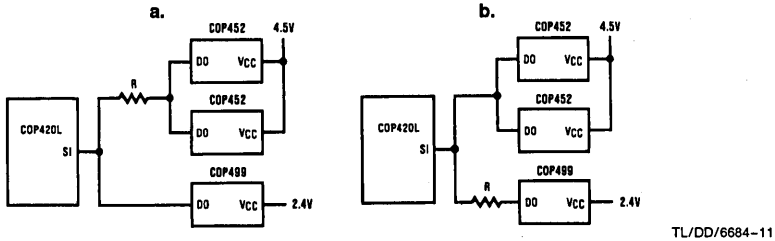
```
WRITE: JSRP   SETUP       ; INITIALIZE, SEE FIGURE 9B
RW1:   XAS                ; SEND COMMAND
RW2:   LD                 ; POSITION Bd
       XDS
RW:    LD
       XAS
       X      3           ; USE REGISTERS 2 AND 1
       LD
       NOP
       XAS
       XIS    3
       JP     RW
       OBD                ; DESELECT THE COP498/499
       JP     FINISH
```

FIGURE 10. COP410L/411L-COP498/499 Special Routine

2. The device does not become dynamic until both CS and CE are high and at least one high level is seen at the SK input. Thus the device may be safely enabled prior to turning on the clock as long as SK is low when the device is enabled.
3. The device must be deselected between instructions. Failure to do so will yield improper operation. The device relies on the select lines changing state in order to clear internal registers. Only one of the select lines on the COP498 needs to go low between instructions.
4. The user must insure that a WREN (write enable) instruction has been performed in order to write to the device memory. The WREN command need be given only once unless the SLEEP feature is used. If $\overline{ON}$ goes high as a result of a SLEEP command, a write disable is automatically performed in order to provide maximum protection to the device memory while the COPS controller is powering up and powering down. As long as $\overline{ON}$ remains high, WRITE and WREN instructions are disabled. Thus when the COPS controller wakes up after previously issuing a SLEEP command, a WREN instruction is required before data can be written to the device.
5. The six bit section of the RAM address register will increment whenever there are clock pulses present when the CS and CE pins are high. Thus the user can position the RAM address register if he wishes by selecting the device, holding the DI pin low and supplying the appropriate number of clocks. Then, without deselecting the device, the user would send the instruction and read or write data. Although possible, this technique is not recommended as it is fairly involved.
6. When using the TSEC command in COP498 with the code as given in *Figure 9,* the master program should test for the accumulator greater than 1 to determine if the timer seconds latch was set. Note again, test for greater than 1; do not test for greater than zero.

NOTE ON MICROWIRE INTERFACE

If the device is connected to a MICROWIRE interface containing other circuits whose DO (data output) pins may produce a signal swing higher than V_{CC} of the device, some protection is needed on the DO pin of the device. This happens when the DO pins of several peripherals powered by different voltages are connected together; e.g., a COP452 at 4.5V with a COP499 at 2.4V. When the DO pin of COP498/499 is externally driven above its power supply voltage, a current will flow into it and this current must be limited to 1 mA. As an example we have two COP452s with a COP420L operating at 4.5V and a COP499 operating at 2.4V. When enabled, the DO pin of a COP452 may swing higher than 2.4V, the power supply voltage of the COP499. One way to limit the current is to use a current limiting resistor of 2 kΩ between the DO pins of the COP452 and the COP499. NOTE: the SI pin of the COPS processor MUST BE A Hi-Z INPUT. Two configurations are possible as shown in *Figure 11.* Note that the resistor between DO and SI will give extra RC delay to the signal going from the DO pin to the SI pin of the COPS processor. Connection B is preferred because the DO signal from COP499 has nearly a whole SK cycle to become valid at SI input before the signal is read by the processor. When a ROMless COPS processor (COP401L/COP402/COP404L) is used for emulation, the circuit shown in *Figure 12* may be used to simulate a Hi-Z input for the SI pin.



TL/DD/6684-11

FIGURE 11. High Voltage Protection on DO pin

Functional Description (Continued)

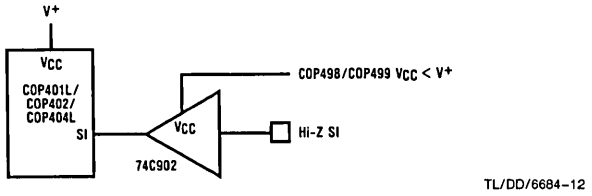


FIGURE 12. Simulating HI-Z SI Input on ROMless Processors

Section 7

Display/Terminal Management Processor (TMP)

Section 7 Contents

National Semiconductor

TMP™

Terminal Management Processor

The TMP (NS405 series) is a single-chip CRT terminal display controller. The TMP is supported by the MOLE™ development system and replaces all the following LSI circuits commonly found in a terminal:

- Microprocessor
- Program ROM
- 64 x 8 RAM
- CRT controller
- DMA controller
- Character generator
- UART
- BAUD rate generator
- Parallel I/O controller
- Timer

The TMP offers complete CRT control over a wide scope of high-density circuit applications including phones, keyboard integration assignments, logic analyzers and more.

The NS455 Terminal Management Processor (TMP) demo board is available for design support.

Highly compact, the TMP board reduces previously necessary board space dramatically while providing 100% emulation of a classic low-end terminal. The board can also be used for TMP evaluation or as a vehicle for designing-in the NS405 device.

The board which is controlled by a preprogrammed NS455, needs only a video monitor, ASCII encoded keyboard, and power supply to provide your complete terminal. Should you wish to write your own program, no problem.

The cross-assembler software provides the capability. The board will execute custom programs through up to 8k of off-chip memory.

The TMP demo board comes complete with operating manual, program source listing, board schematic, board layout, and all necessary connectors.

When you're ready to design your own TMP system, turn to National's MOLE development system. By using this system-comprised of brain board, personality board and software—you bring dedicated development support to the TMP chip, making design-in extremely fast and simple.

National Semiconductor

PRELIMINARY

NS405-Series Display/Terminal Management Processor (TMP)

General Description

The NS405 is a CRT terminal controller on a chip. It is a microcomputer system which replaces the following LSI circuits commonly found in a CRT data terminal:

- Microcomputer
- Baud Rate Generator
- CRT Controller
- Interrupt Controller
- DMA Controller
- Parallel I/O Controller
- Character Generator
- Timer
- UART

In addition the NS405 includes powerful attribute logic, two graphics display modes, and fast video output circuits.

The NS405 is primarily intended for use in low-cost terminals, but contains many features which make it a superior building block for "smart" terminals and word processing systems.

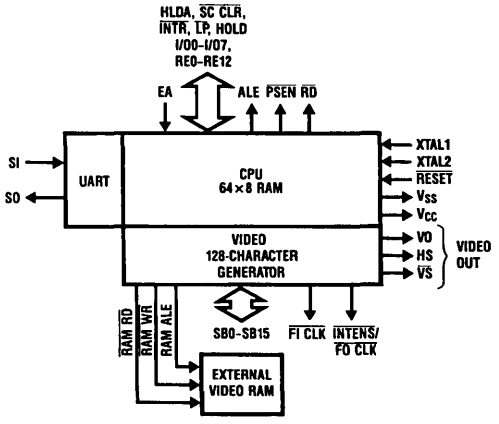
The NS405 interfaces easily to the display monitor, keyboard, display memory, and I/O ports. The architecture and instruction set are derived from the 8048-series microcontrollers. The instruction set has been enhanced and the architecture tailored to allow the NS405 CPU to efficiently manage a large display memory and an extensive interrupt environment.

The TMP can be used to easily and inexpensively add a display to many systems where it was previously impractical, it is not limited to terminal applications.

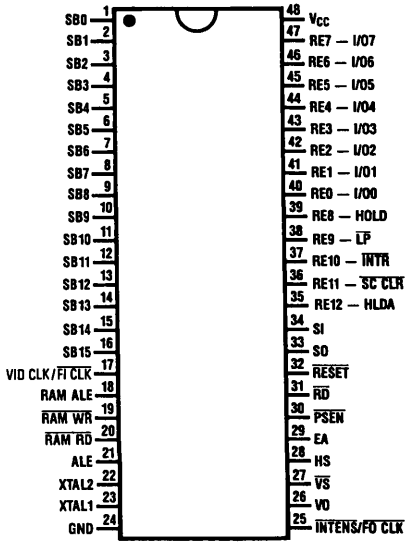
Features

- Enhanced 8048 instruction set and architecture
- Up to 8k x 8 ROM external with ROM expand bus
- On-board RAM 64 x 8
- Programmable display format
- On-board video memory management unit
- 16-bit bidirectional display memory bus (direct video and attribute RAM interface)
- Built-in timer
- Real-time clock (may be programmed for 1 Hz)
- Video control signals
- Eight independent attributes
- Pixel and block graphics display modes
- Programmable cursor characteristics
- Programmable CRT refresh rate
- Light pen feature
- UART, programmable baud rate up to 19.2k baud
- Character generator (128 characters 7 x 11 max)
- Single 5-volt supply @ 110 mA (typ)
- Up to 18 MHz video dot rate (12 MHz CPU clock)
- 48-pin package
- 8-bit parallel I/O port (multiplexed with external ROM)
- Extensive I/O expansion capabilities
- Up to 64k by 8 or 16 video RAM

Block and Connection Diagrams



TL/DD/5526-1



Top View

TL/DD/5526-2

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}*	−0.5V to +7.0V
Power Dissipation	1.5W
ESD	2000V

*EA, SI and VSYNC may be subjected to V_{SS} + 15V.

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operations should be limited to those conditions specified under DC Electrical Characteristics.*

DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{IL1}	Input Low Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)		−0.5	0.8	V
V_{IH1}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)		2.0	V_{CC}	V
V_{IL2}	Input Low Voltage (XTAL1, XTAL2, $\overline{RESET}$)		−0.5	0.6	V
V_{IH2}	Input High Voltage (XTAL1, XTAL2, $\overline{RESET}$)		3.8	V_{CC}	V
V_{OL}	Output Low Voltage (All Except $\overline{INTENS}$, VO)	I_{OL} = 2.0 mA		0.4	V
V_{OH}	Output High Voltage (All Except $\overline{INTENS}$, VO)	I_{OH} = −125 μA	2.4	V_{CC}	V
V_{OL}	Output Low Voltage ($\overline{INTENS}$, VO)	I_{OL} = 5.0 mA		0.4	V
V_{OH}	Output High Voltage ($\overline{INTENS}$, VO)	I_{OH} = −500 μA	2.4		V
I_{IL}	Input Leakage Current (EA, $\overline{INT}$, SI)	$V_{SS} \le V_{IN} \le V_{CC}$		±10	μA
I_{OL}	Output Leakage Current (ROM Expand Bus, High Impedance State)	$V_{CC} \ge V_{IN} \ge V_{SS}$ + 0.45		±10	μA
I_{OL}	Output Leakage Current (System Bus, High Impedance State)	$V_{CC} \ge V_{IN} \ge V_{SS}$ + 0.45		±100	μA
I_{CC}	Total Supply Current	T_A = 25°C		150	mA

AC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified

Symbol	Parameter	Min	Max	Units
CPU AND ROM EXPAND BUS TIMING				
F_{XTAL}	Crystal Frequency	3	18	MHz
F_{CPU}	CPU Frequency	3	12	MHz
t_{CY}	CPU Cycle Time	1.25	7.5	μs
t_{DF}	Video Dot Time	55.5	333.3	ns
t_{LL}	ALE Pulse Width (Note 1)	125		ns
t_{AL}	Address Setup to ALE (Note 1)	55		ns
t_{LA}	Address Hold from ALE (Note 1)	40		ns
t_{CC}	Control Pulse Width $\overline{PSEN}$, $\overline{RD}$ (Note 1)	250		ns
t_{DR}	Data Hold (Notes 1, 4)	0	100	ns
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to Data In (Note 1)		220	ns
t_{AD}	Address Setup to Data In (Note 1)		360	ns
t_{AFC}	Address Float to $\overline{RD}$, $\overline{PSEN}$ (Notes 1, 5)	0		ns
t_{CAF}	$\overline{PSEN}$ to Address Float (Notes 1, 5)	−10	+10	ns
t_{DAL}	Data Setup to ALE (RE0–7, 11, 12) (Note 1)	55		ns
t_{ALD}	Data Hold from ALE (RE0–7, 11, 12) (Note 1)	40		ns
t_{CIS}	Control Input Setup to ALE (RE8, 9, 10) (Note 1)	240		ns
t_{CIH}	Control Input Hold from ALE (RE8, 9, 10) (Notes 1, 4)	75	125	ns

AC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified (Continued)

Symbol	Parameter	Min	Max	Units
SYSTEM BUS TIMING				
t_{EL}	RAM ALE Low Time (Note 1)	250		ns
t_{EH}	RAM ALE High Time (Note 1)	100		ns
t_{AS}	Address Setup to RAM ALE (Note 1)	20		ns
t_{AH}	Address Hold from RAM ALE (Note 1)	10		ns
t_{RR}	$\overline{\text{RAM RD}}$ Width (Note 1)	210		ns
t_{AR}	Address Setup to $\overline{\text{RAM RD}}$ (Note 1)	80		ns
t_{RRD}	Data Access from $\overline{\text{RAM RD}}$ (Note 1)		140	ns
t_{RDR}	Data Hold from $\overline{\text{RAM RD}}$ (Notes 1, 4)	0	60	ns
t_{WFI}	$\overline{\text{FIFO In Clock}}$ Width (Note 1)	210		ns
t_{WW}	$\overline{\text{RAM WR}}$ Strobe Width (Note 1)	130		ns
t_{AW}	Address Setup to $\overline{\text{RAM WR}}$ (Note 1)	120		ns
t_{DW}	Data Setup to $\overline{\text{RAM WR}}$ (Note 1)	10		ns
t_{WD}	Data Hold from $\overline{\text{RAM WR}}$ (Note 1)	20		ns
VIDEO TIMING				
t_{DF}	Dot Period = $\frac{1}{f_c}$ (Note 1)	55		ns
t_{VID}	Video Blank Time (Note 1)	5	15	ns
t_{VI}	Skew, $\overline{\text{Intensity}}$ to Dot 0 (Note 1)	−15	15	ns
t_{FOV}	$\overline{\text{FIFO Out Clock}}$ to Dot 0 (Note 1)		15	ns
t_{WFOH}	$\overline{\text{FIFO Out Clock}}$ Width High (Note 1, Note 2)	55	165	ns

*1/3 CPU cycle.

**1 Dot time is 55 ns.

Note 1: Control outputs C_L = 80 pF; ROM Expand Bus outputs C_L = 150 pF; System Bus outputs C_L = 100 pF; V_{OUT} & $\overline{\text{INTENS}}$ outputs C_L = 50 pF; F_{XTAL} = 18 MHz; F_{CPU} = 12 MHz. XTAL1 & XTAL2 driven externally per *Figure 12b* with 50% duty cycle.

Note 2: $\overline{\text{FO CLK}}$ duty cycle is shown above.

Note 3: Hold request is latched. It is honored at the start of the next vertical retrace.

Note 4: Max spec. listed for user information only, to prevent bus contention. Maximum value not tested.

Note 5: Not tested.

Character Cell Width	$\overline{\text{FIFO Out}}$ HIGH	$\overline{\text{FIFO Out}}$ LOW
6	1 dot	5 dots
7	2 dots	5 dots
8	2 dots	6 dots
9	3 dots	6 dots
10	3 dots	7 dots

Input Hold Times

T_A = 25°C, V_{CC} = +5V ±10%, V_{SS} = 0V

Input	Min Active Time
Reset	50 ms (power up) 5 CPU Cycles (after power up)
$\overline{\text{External Interrupt}}$	2 CPU Cycle
$\overline{\text{Light Pen}}$	1 CPU Cycle
I/O Input	1 CPU Cycle
Hold Request	1 CPU Cycle (Note 3)

FIFO

Fall through should not be greater than 4 character times (character time = $1/f_{XTAL}$ × #dots/cell).

Throughput rate must be at least the character rate (character rate = 1/character time).

Capacitance $T_A = 25°C, V_{CC} = V_{SS} = 0V$

Symbol	Parameter	Test Conditions	Min	Max	Units
C_{IN}	Input Capacitance	F_C = 1 MHz (Note 5)		10	pF
C_{OUT}	Output and Reset	Unmeasured Pins Returned to V_{SS} (Note 5)		20	pF

AC Electrical Characteristics in CPU Cycle Time

CPU AND ROM EXPAND BUS TIMING (FOR REFERENCE ONLY)

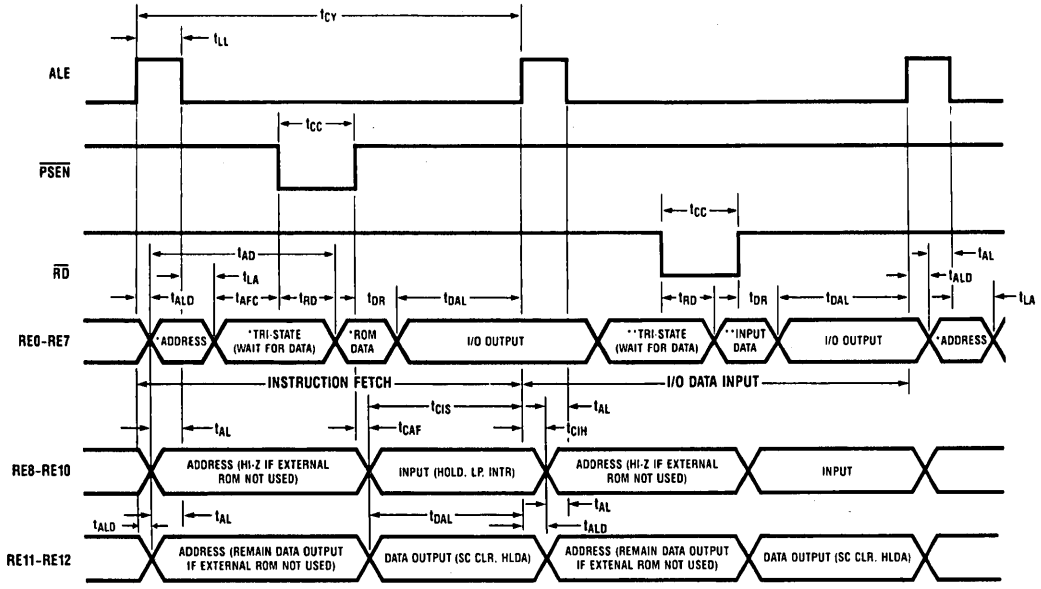
Symbol	Parameter		Typ
t_{LL}	ALE Pulse Width		14 $t_{CY/60}$
t_{AL}	Address Setup to ALE		8 $t_{CY/60}$
t_{LA}	Address Hold from ALE		6 $t_{CY/60}$
t_{CC}	Control Pulse Width	$\overline{PSEN}$ $\overline{RD}$	24 $t_{CY/60}$ 36 $t_{CY/60}$
t_{CY}	CPU Cycle Time		$60\ t_{CY/60} = 15/f_{CPU} = \frac{15}{f_{XTAL} \div 1 \text{ or } \div 1.5}$
t_{DR}	Data Hold		$-2\ t_{CY/60}$
t_{RD}	Control Pulse to Data In	$\overline{PSEN}$ $\overline{RD}$	18 $t_{CY/60}$ 30 $t_{CY/60}$
t_{AD}	Address Setup to Data In		32 $t_{CY/60}$
t_{AFC}	Address Float to	$\overline{PSEN}$ $\overline{RD}$	2 $t_{CY/60}$ 2 $t_{CY/60}$
t_{CAF}	PSEN to Address Float		0 $t_{CY/60}$
t_{DAL}	Data Setup to ALE	RE0–7 RE8–10 RE11–12	6 $t_{CY/60}$ $-2\ t_{CY/60}$ 16 $t_{CY/60}$
t_{ALD}	Data Hold from ALE	RE0–7 RE8–12	2 $t_{CY/60}$ 6 $t_{CY/60}$

SYSTEM BUS TIMING (FOR REFERENCE ONLY)

Symbol	Parameter	Ticks	
		Min	Max
t_{EL}	RAM ALE Low Time	14 $t_{CY/60}$ − 42 ns	
t_{EH}	RAM ALE High Time	6 $t_{CY/60}$ − 25 ns	
t_{AS}	Address Setup to RAM ALE	4 $t_{CY/60}$ − 60 ns	
t_{AH}	Address Hold from RAM ALE	2 $t_{CY/60}$ − 40 ns	
t_{RCY}	Read or Write Cycle Time		
t_{RR}	$\overline{RAM\ RD}$ Width	12 $t_{CY/60}$ − 40 ns	
t_{AR}	Address Setup to $\overline{RAM\ RD}$	6 $t_{CY/60}$ − 45 ns	
t_{RRD}	Data Access from $\overline{RAM\ RD}$		10 $t_{CY/60}$ − 70 ns
t_{RDR}	Data Hold from $\overline{RAM\ RD}$		
t_{WFI}	$\overline{FIFO\ In\ Clock}$ Width	12 $t_{CY/60}$ − 40 ns	
t_{WW}	$\overline{RAM\ WR}$ Strobe Width	8 $t_{CY/60}$ − 27 ns	
t_{AW}	Address Setup to $\overline{RAM\ WR}$	10 $t_{CY/60}$ − 90 ns	
t_{DW}	Data Setup to $\overline{RAM\ WR}$	2 $t_{CY/60}$ − 30 ns	
t_{WD}	Data Hold from $\overline{RAM\ WR}$	2 $t_{CY/60}$ − 20 ns	

Timing Waveforms

ROM Expand Bus Timing
(In Port Instruction Is Shown)

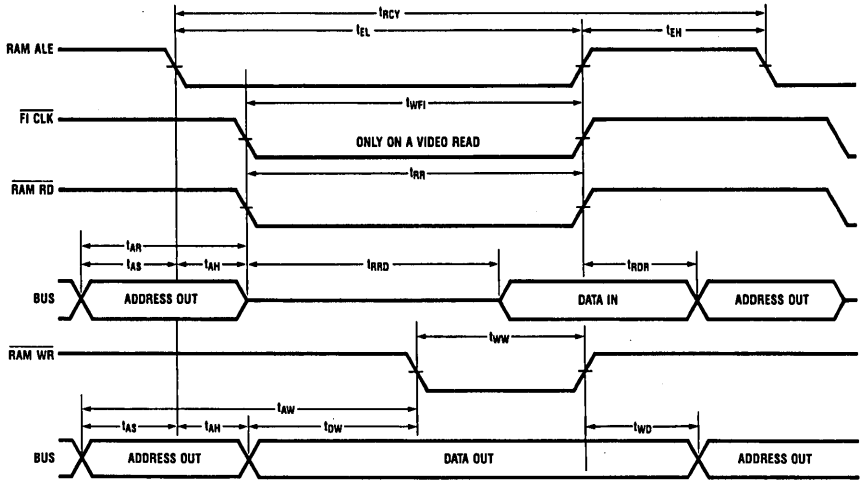


TL/DD/5526-3

*Remain I/O OUTPUT if External ROM not used.

**I/O Data input or 2nd ROM byte of 2 byte instruction. Otherwise remain I/O OUTPUT.

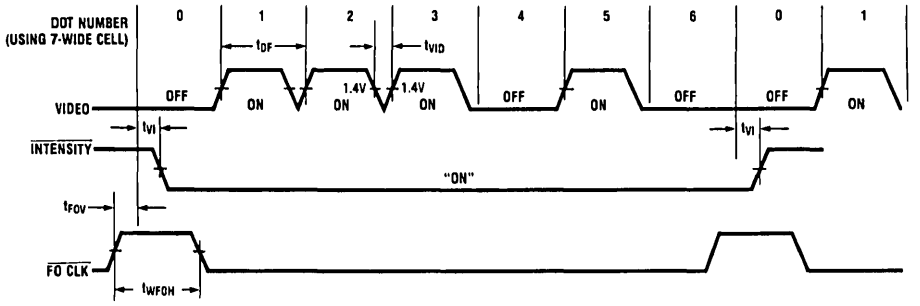
System Bus Timing



TL/DD/5526-4

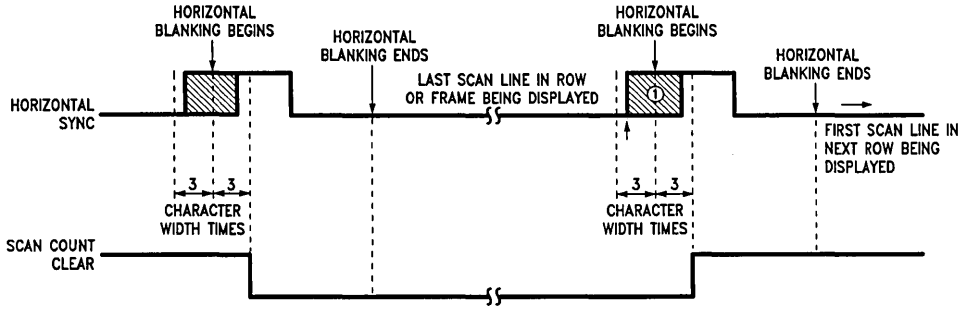
Timing Waveforms (Continued)

Video Timing



TL/DD/5526-5

Scan Count Clear Timing

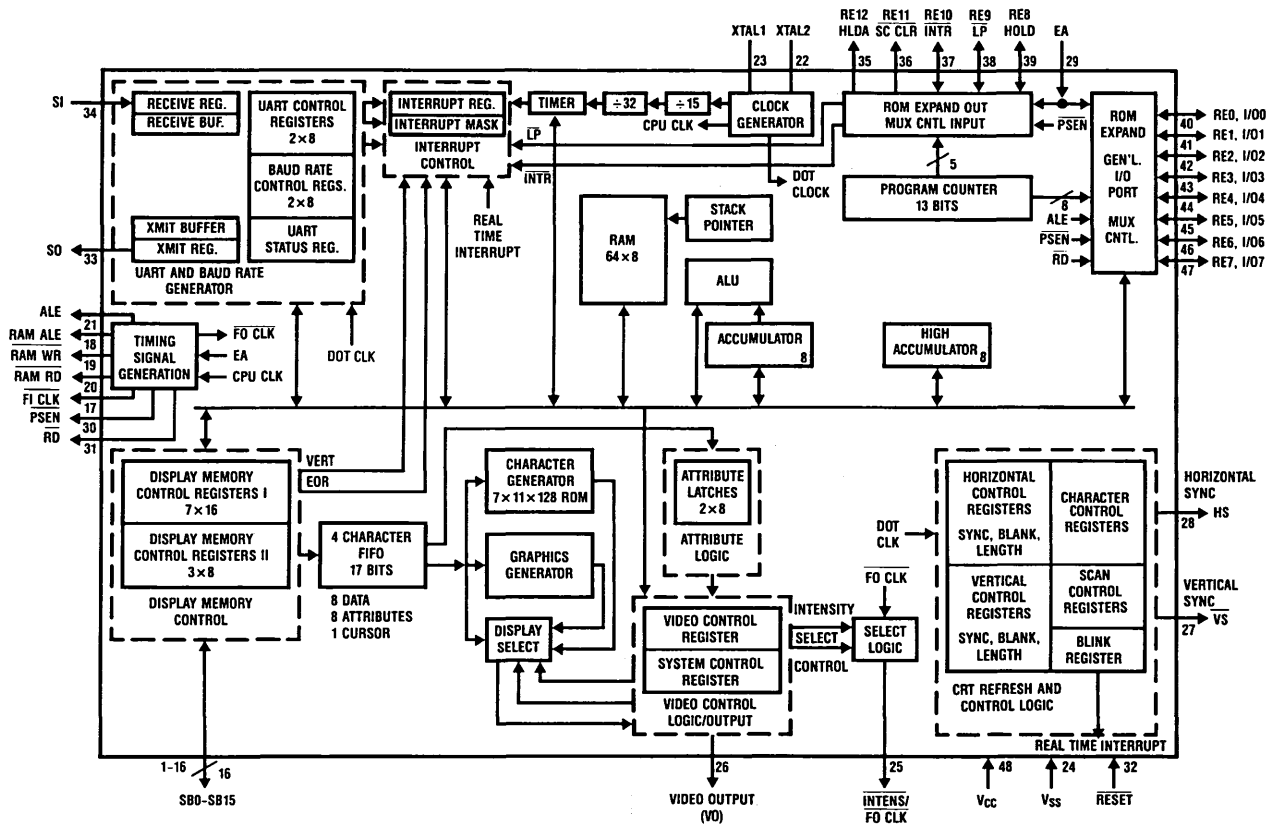


TL/DD/5526-6

For external character generation this edge is used to clock CLEAR into scan line counter. The edge must come before Scan Count Clear goes away, but not before the video controller has brought in all necessary display information for the last scan line.

Block Diagram

NS405-Series Detailed



TL/DD/5526–7

1.0 Functional Pin Descriptions

1.1 SUPPLIES

Pin	Name	Function
48	V_{CC} — Power	5V ±10%
24	V_{SS} — Ground Reference	

1.2 INPUT SIGNALS

23, 22	XTAL1, XTAL2 — Crystal 1, 2:	Crystal connections for clock oscillator (3–18 MHz).
29	EA — External Access:	Pull HIGH (V_{IH_2})
32	$\overline{\text{RESET}}$	An active low input that initializes the processor. The $\overline{\text{RESET}}$ input is also used for internal ROM verification.
34	SI — Serial Input:	Drives receiver section of UART (true data).

1.3 OUTPUT SIGNALS

33	SO — Serial Output:	Driven by transmitter section of UART (true data).
21	ALE — Address Latch Enable:	ROM address is available on the ROM Expand Bus and may be latched on the falling edge of ALE. Port output data may be latched on the rising edge of ALE. ALE pulses are always present, even if EA is tied low.
30	$\overline{\text{PSEN}}$ — Program Store Enable:	Enable external ROM output drivers when low. $\overline{\text{PSEN}}$ is idle (high) when the CPU fetches from internal ROM.
31	$\overline{\text{RD}}$ — Read Port Data:	Accept Port input data on ROM Expand Bus RE0–RE7 while low. ROM Expand Bus is in high impedance state while $\overline{\text{RD}}$ is low.
28	HS — Horizontal Sync	The rising edge of HS is controlled by the Horizontal Sync Begin Register and the falling edge is controlled by the Horizontal Sync End Register. HS is disabled (low) if bit 5 of the Video Control Register = 0.
27	$\overline{\text{VS}}$ — Vertical Sync Output:	The falling edge of $\overline{\text{VS}}$ is controlled by the Vertical Sync Begin Register and the rising edge is controlled by the Vertical Sync End Register. $\overline{\text{VS}}$ is at TRI-STATE if bit 5 of the Video Control Register = 0.
26	VO — Video Output:	High = beam on, low = beam off. VO is disabled (low) if bit 5 of the Video Control Register = 0.
25	$\overline{\text{INTENS}}$/$\overline{\text{FO CLK}}$	(Shared pin) $\overline{\text{INTENS}}$ Signal under attribute control may be used to switch the bistable brightness of display characters. $\overline{\text{FIFO Out Clock}}$ may be used to clock data from an external FIFO in synchronism with data from the internal FIFO. Both CANNOT be used simultaneously.
17	VID CLK/$\overline{\text{FI CLK}}$ — Video Dot Clock Out/ FIFO IN CLOCK	(Shared pin) The rising edge of the Video Dot Clock may be used to clock the data out of the video output pin. FIFO In Clock may be used to clock data from an extended attribute RAM into an external FIFO in synchronism with the data loaded into the internal FIFO. Both CANNOT be used simultaneously.
18	RAM ALE — RAM Address Latch Enable:	RAM address is available on the System Bus and may be latched on the falling edge of RAM ALE. Only operational when Display RAM accesses being performed. Otherwise high.
20	$\overline{\text{RAM RD}}$ — RAM Read:	Enable display RAM data onto the System Bus when $\overline{\text{RAM RD}}$ is low.
19	$\overline{\text{RAM WR}}$ — RAM Write:	Data to RAM is available on the System Bus and may be written at the rising edge of $\overline{\text{RAM WR}}$.

1.4 BUS — I/O

1–8	SB0–SB7 — System Bus 0–7:	Display RAM address is output while RAM ALE is high and may be latched on the falling edge of RAM ALE. System Bus accepts data input while $\overline{\text{RAM RD}}$ is low and outputs data while $\overline{\text{RAM WR}}$ is low.
9–16	SB8–SB15 — System Bus 8–15:	Normally, Display RAM address is output and held on these pins for the full read or write cycle. However, if bit 4 of the System Control Register is set, these pins function bidirectionally like SB0–SB7 to allow 16-bit data words for attribute operation.
35–47	RE0–12 — ROM Expand Bus 0–12:	Used for program ROM expansion as described below. Time multiplexed with I/O port and system control signals. I/O port and system control signals only if no external ROM used.
40–47	RE0–RE7	Low order ROM address is output and may be latched on the falling edge of ALE. Enable ROM data to this Bus when $\overline{\text{PSEN}}$ is low. Enable I/O port input data to the Bus when $\overline{\text{RD}}$ is low. Use the rising edge of ALE to latch port output data.

1.0 Functional Pin Description (Continued)

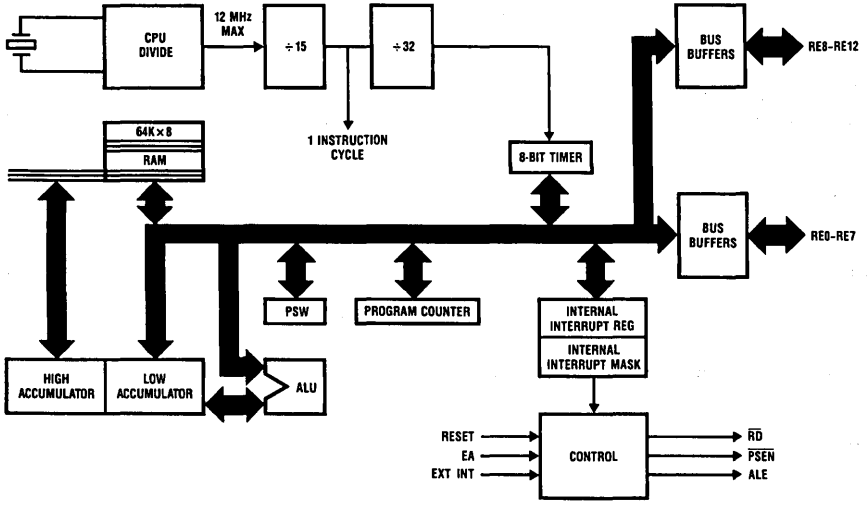
Pin	Name	Function
39–35	RE8–RE12	Five most significant bits of the ROM address are output during ALE and remain stable until data is read in during $\overline{PSEN}$. These pins are multiplexed with the HLDA, $\overline{INTR}$, $\overline{LP}$, $\overline{SC\ CLR}$, and HOLD signals.
37	$\overline{INTR}$ — Interrupt: RE10	An active low input that interrupts the processor if the external interrupt is enabled. Because it shares a pin with RE10, $\overline{INTR}$ may be driven directly only if no external ROM is used (EA is low). Otherwise must be driven through a 3.9k resistor.*
38	$\overline{LP}$ — Light Pen Interrupt: RE9	An active low input that interrupts the processor if internal interrupts are enabled and bit 5 in the Interrupt Mask Register is set. Because it shares a pin with RE9, $\overline{LP}$ may be driven directly only if EA is low. Otherwise, must be driven through a 3.9k resistor.*
39	HOLD — HOLD request: RE8	When high, requests that the NS405 enter the Hold mode. When in the Hold mode the System Bus will be in a high impedance state. The Hold mode is granted at the beginning of the next vertical retrace. Because it shares a pin with RE8, HOLD may be driven directly only if EA is low. Otherwise, must be driven through a 3.9k resistor.*
35	HLDA — Hold Acknowledge: RE12	This output is asserted in response to Hold and provides handshake capability with another processor (active high). For more detailed information see Section 3.0 Slave Processing. Because HLDA shares a pin with RE12, the HLDA state is preset only during the interval preceding the rising edge of ALE. However, if no external ROM is used, HLDA is a steady state output and need not be latched externally.
36	$\overline{SC\ CLR}$ — Scan Count Clear: RE11	This output clears an external scan counter when used with an external character generator. It is a low going pulse which occurs during the horizontal retrace preceding the first scan line of each character row. Because $\overline{SC\ CLR}$ shares a pin with the RE11, the correct $\overline{SC\ CLR}$ state is present only during the interval preceding the rising edge of ALE. However, if no external ROM is used, $\overline{SC\ CLR}$ is a steady state output and need not be latched externally.

*Unused control inputs must be terminated

2.0 Functional Description

2.1 CPU

The CPU of the NS405 is patterned after the 8048 single chip microcomputer (see *Figure 1*).



TL/DD/5526–8

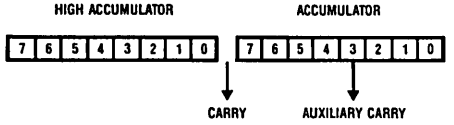
FIGURE 1. NS405 Series CPU Block Diagram

2.0 Functional Description (Continued)

2.1.1 Accumulator — High Accumulator

In addition to the regular 8-bit Accumulator, there is an 8-bit High Accumulator extension to facilitate the 16-bit operations required for display memory management. The HACC/ACC pair is usually used in conjunction with the 16-bit RAM pointer registers (RA, R0 and RB, R1, CURSOR, HOME, BEGD and ENDD) to effect video data transfers. In addition, external attribute memory is loaded in a 16-bit transfer operation. Any instruction which causes a carry or borrow out of the low accumulator will affect the high accumulator (see *Figure 2*).

Auxiliary carry is used only when converting the accumulator contents from binary to BCD (binary coded decimal) using the DA A instruction. The auxiliary carry flag can be cleared by moving a zero into bit 6 of the program status word.

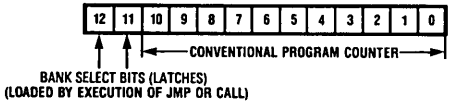


TL/DD/5526-9

FIGURE 2. CPU Accumulator

2.1.2 Program Counter (PC)

The Program Counter is a 13-bit wide register which provides program addressing for the CPU. The lower 11 bits operate like a conventional program counter while the upper 2 bits are actually latches. These 2 latches are automatically loaded from the bank select flip-flops (PSW bits 3, 4) whenever a JMP or CALL instruction is executed. The bank select flip-flops in turn are only modified upon the execution of a Select Memory Bank Instruction or modification of the PSW (see *Figure 3*).



TL/DD/5526-10

FIGURE 3. TMP Program Counter

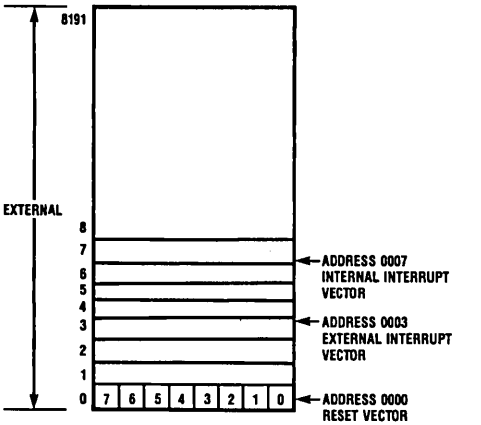
2.1.3 Program Memory

Memory is subdivided into 2k banks with accesses limited to the currently selected bank unless a Bank Change sequence has been executed. Upon reaching the end of a memory bank, the program counter will wrap around and point to the beginning of the current bank.

Each bank is further subdivided into pages of 256 bytes each, with 8 pages in every bank. The conditional JUMP instructions are restricted to operate within the memory page that they reside in.

Because of the sequence which the CALL instruction executes when pushing and loading the PC, it is possible to easily call and return from subroutines located in different memory banks (see *Figure 4*).

Upon executing an RET or RETR instruction for a call from one memory bank into another, a SEL MBx instruction should be excuted to restore the memory bank select flip-flops to their original bank. However, no SEL MBx is needed after an interrupt since the flip-flops were never modified.



TL/DD/5526-11

FIGURE 4. Program Memory Map

2.1.4 Program Status Word Bit Assignments

Bit Position	Contents
0	Stack Pointer Bit, S0
1	Stack Pointer Bit, S1
2	Stack Pointer Bit, S2
3*	Memory Bank Select Bit 0
4*	Memory Bank Select Bit 1
5*	Register Bank Select Bit (0 = Bank 0, 1 = Bank 1)
6*	Auxiliary Carry. A carry from Bit 3 to Bit 4 generated by an add operation. Used only by the decimal adjust (DA A) instruction.
7*	Carry. A bit indicating the preceding operation resulted in an overflow or an underflow from the 8-bit accumulator.

***Note 1:** Bits 3 through 7 are saved on the stack by subroutine calls or interrupts. Bits 3 and 4 are restored upon execution of an RET instruction, whereas all 5 bits are restored by RETR.

Note 2: F0 is not saved on the stack (as in an 8048).

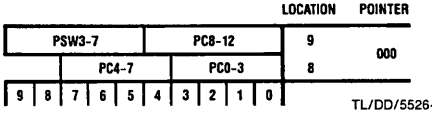
Note 3: Bits 0-5 cleared on a RESET.

2.1.5 Stack Pointer (SP)

The stack pointer is an independent 3-bit counter which points to designated locations in the internal RAM that holds subroutine return parameters. The stack itself is located in RAM locations 8-23 (see *Figure 5*).

Each entry in the stack takes up two bytes and contains both the PC and status bits. When reset to zero, the stack pointer actually points to locations 8 and 9 in RAM. Since the stack pointer is a simple up/down counter, an overflow will cause the deepest stack entry to be lost (the counter overflows from 111 to 000 and underflows from 000 to 111).

Note: If the level of subroutine nesting is less than eight (8), the unneeded stack locations may be used as RAM.



TL/DD/5526-12

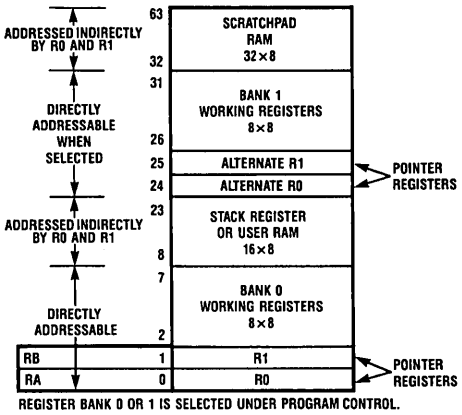
Note: The odd numbered RAM bytes in the stack area have two (2) extra bits to allow for storage of the bank select switch bits. This feature allows interrupt routines and subroutines to be located outside the current 2k program memory bank.

FIGURE 5. Typical Stack Composition

2.0 Functional Description (Continued)

2.1.6 Data Memory (On-Chip RAM)

The data memory nominally consists of 64 8-bit locations and is utilized for working registers, the subroutine stack, pointer registers and scratch pad. There are two sets of working/pointer registers (R0–R7) which are selected by the Select RAM Bank instruction. The stack area is located in locations 8–23. Locations 32–63 contain the scratch pad memory. To facilitate 16-bit Video Memory Management there are two 8-bit extension registers (RA and RB) which are associated with the R0 and R1 registers respectively of whichever RAM bank is currently selected (see *Figure 6*). i.e., There is only one RA register and only one RB register.

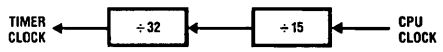


TL/DD/5526–13

FIGURE 6. RAM Memory Map

2.1.7 Timer

The On-Board Timer is an 8-bit up counter which sets the Timer Overflow Flag and generates an internal interrupt (if enabled) whenever it overflows from FF to zero. The Timer may be stopped, started, loaded and read from by the CPU. The Timer clock is derived from the CPU clock as shown in *Figure 7*. Whenever a Start Timer instruction is executed the ÷32 is initialized to its zero state to insure a full count measurement. After overflow the timer keeps counting until the next FF to zero overflow at which time the overflow flag will be set and another interrupt generated. The overflow flag can only be reset through the JTF and JNTF instructions.



TL/DD/5526–14

FIGURE 7. Timer Clock Generation

2.1.8 Interrupts

The interrupt circuitry handles two generic classes of interrupt conditions called Internal and External. Either class has its own master control which can be activated through software enable and disable instructions. On an interrupt service the currently executing instruction is completed, then two CPU cycles are used as the program counter and bits 3–7 of the PSW are pushed onto the stack and stack pointer is incremented.

Then the interrupt vector address (3 or 7) is loaded into the PC and service started. Whenever an interrupt condition is being serviced all other interrupts of either class are locked out until a RETR instruction is executed to conclude interrupt service. If both an external and internal interrupt arrive at the same time, the external interrupt is recognized first.

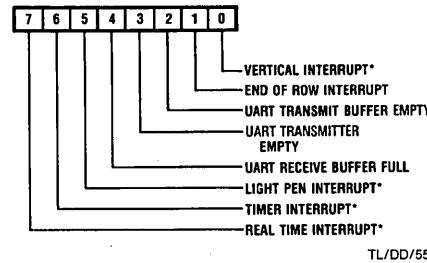
2.1.8.1 External Interrupt

The External Interrupt consists solely of the shared $\overline{\text{INTR}}$/RE10 pin. External interrupts on this pin will be detected if the setup and hold times as shown in the timing diagrams are met. This pin is a level sampled interrupt which means that as long as the pin is low during the sampling window an interrupt will be generated. In addition, the $\overline{\text{INTR}}$ pin is the only external pin whose logic state can be directly tested through software.

2.1.8.2 Internal Interrupts

The Internal Interrupts consist of seven internal operational conditions plus the light pen arranged in an 8-bit wide register as shown in *Figure 8*. Activation of an internal interrupt condition causes a corresponding register bit to be set, *Figure 9*. Each internal interrupt may be individually masked out through the Interrupt Mask register which has the same bit assignments as the Interrupt register and can be loaded from the accumulator. A zero in the Interrupt Mask register inhibits the interrupt and a one enables it. Further interrupt processing is as shown. To determine which of the eight internal conditions caused the interrupt the CPU must read the Interrupt register into the accumulator. To acknowledge receipt of the interrupt certain bits are automatically cleared on a read while others are reset upon service of the particular interrupt.

The conditions under which each of the interrupts are generated and cleared are as follows:



TL/DD/5526–16

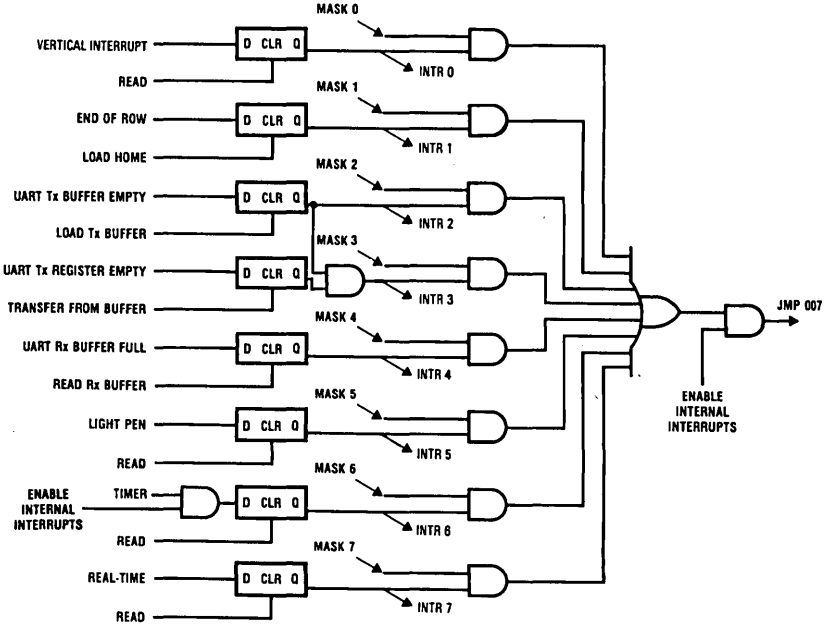
Note: The interrupt flags indicated by an asterisk (*) are cleared when the Interrupt Register is read.

FIGURE 8. Internal Interrupt Register

Bit

0 Vertical Interrupt—Generates an interrupt at the end of the display row designated by the Vertical Interrupt Register. Interrupt bit cleared on a CPU read of the interrupt register. If VIR > Vertical Length Register no interrupt will be generated.

2.0 Functional Description (Continued)



TL/DD/5526–15

FIGURE 9. Internal Interrupt Processing

Bit

1 End of Row Interrupt—Generates an interrupt at the end of each display row when the Current Row Start Register is updated for the next row. Used in conjunction with the Row Sequencing Control Bit (5) in the System Control Register to implement Row Pointer Look-Up Tables and Horizontally Split Screens. Interrupt bit cleared on a CPU write to the Home Register. Does not generate interrupts for those rows blanked during vertical blanking.

2 UART Transmit Buffer Empty—Generates an interrupt when the Transmit Buffer empties out after dumping a character into the Transmit Shift Register. Interrupt bit cleared on a CPU write to the Transmit Buffer.

3 Transmitter Empty—Generates an interrupt when BOTH the Transmit Buffer and Transmit Shift Register are empty. The interrupt bit is cleared when the CPU loads the transmit buffer.

4 UART Receiver Buffer Full—Generates an interrupt when the Receiver Buffer fills up with a character from the Receive Shift Register. Interrupt bit cleared on a CPU read of the Receiver Buffer.

5 Light Pen Interrupt—Generates an interrupt on each falling edge detected on the shared $\overline{LP}$/RE9 pin. Since only falling edges generate interrupts and the input is sampled each CPU Cycle, a high level must be sampled between falling edges in order to be considered a new interrupt. This interrupt is used to latch the light pen position registers. For further information see Light Pen Description. Interrupt bit cleared on a CPU read of the interrupt register.

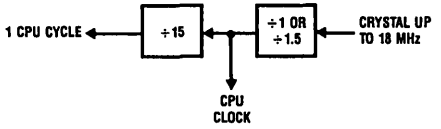
Bit

6 Timer Interrupt—Generates an interrupt when the internal 8-bit Timer overflows from FF to 00. Interrupt bit cleared on a CPU read of the interrupt register.

7 Real-Time Interrupt—Generates interrupts at a software programmable frequency that is generally in the Hertz range. (See CPU Clock Generation.) Thus permitting the implementation of a real-time clock or timer. Interrupt bit cleared on a CPU read of the interrupt register.

2.1.9 Clock Generation

All chip clocks are derived from the one external crystal connected between pins 22 and 23. This master clock also doubles as the video dot clock. The crystal frequency is constrained to lie within the range of 3 to 18 MHz. The CPU clock is derived from the crystal clock by either using it directly or by dividing down by a factor of 1.5 *(Figure 10)*.



TL/DD/5526–17

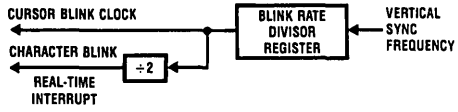
FIGURE 10. CPU Clock Generation

The choice is software programmable through bit 0 in the System Control Register. The exact selection is made in consideration of the fact that the CPU clock must lie within the range of 3 to 12 MHz. In addition, the choice of divide by modes will also impact the display character cell width due to the nature of the video controller. Specifically with ÷1.5

2.0 Functional Description (Continued)

the cell width must be ≥ 8 dots wide whereas with ÷1 the cell width must be ≥ 6 dots wide.

The low clock rates necessary to implement Cursor Blinking, Character Blinking and the Real-Time Interrupt are derived by passing the vertical sync frequency through a 5-bit Blink Rate Divisor Register, *(Figure 11)*. The resultant frequency is used as the Cursor Blink Clock. This clock is then further divided by 2 to yield the Character Blink and Real-Time Interrupt Clocks. For example, to get a 1 Hz real time interrupt, with a 60 Hz system, set the 5 bit Divisor Register to 30 in order to yield a 2 Hz signal which is then divided by 2.

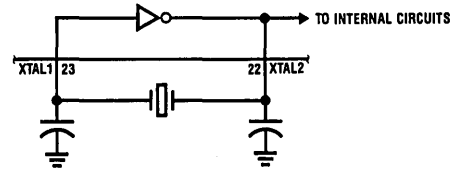


TL/DD/5526–18

FIGURE 11. Blink Clock Generation

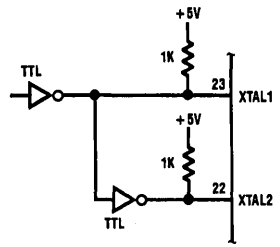
2.1.10 Oscillator Operation

The on-board oscillator circuit consists of a phase inverter which, when used with an external parallel resonant tank, *(Figure 12a)*, will yield the required oscillator clock. Crystals should be specified for AT cut and parallel resonant operation with the desired load capacitance (typically 20 pF). If one desires to externally generate the clock and input it to the chip, he may do so by driving XTAL1 (pin 23) and XTAL2 (pin 22) as shown in *Figure 12b*.



TL/DD/5526–19

FIGURE 12a. TMP Oscillator



TL/DD/5526–20

Note: Use AS TTL devices if faster than 12 MHz.

FIGURE 12b. External Oscillator Mode

2.2 DISPLAY MEMORY CONTROLLER

The video display data resides in the external Video Memory which is managed by the Display Memory Controller (DMC) through the System Bus. Either the CPU or the Video Controller may access the display memory by presenting its requests to the DMC. A maximum of three Video Memory accesses (Reads or Writes) can be performed by the DMC during each CPU instruction execution cycle. Because the CPU can access the Video Memory, one may expand CPU I/O or data memory by memory mapping into the Video Memory space. Up to 64k locations may be addressed over the 16-bit System Bus. Data word widths may be 8 or 16 bits depending upon whether external character attribute selection is used. The actual bus multiplexing mode is controlled by bit 4 in the System Control register. The Video Controller has the highest priority in obtaining Video Memory accesses with the CPU getting in on a space available basis. If all memory accesses are being taken by the Video Controller (rarely), the CPU is put into a wait state should it try to access video memory. To ease accessing requirements and boost throughput the Video Controller utilizes a 4-level data FIFO which is normally kept full of display data.

2.2.1 Display Memory Control Registers

In order to facilitate the management of video data for such features as a Screen scroll, memory paging and row lookup the DMC utilizes a number of registers which address the video RAM space. Each of these pointers is 16 bits wide and writable or readable from the 16-bit HACC/ACC pair as the case may be. There are 2 video data accessing modes as determined by bit 5 in the SCR, Sequential and Table Lookup. The functions of the pointer registers vary depending upon the accessing mode selected. Their designators are:

HOME = Home address register. Read and write.

BEGD = Beginning of diplay RAM. Write only.

ENDD = End of display RAM. Write only.

CURS = Cursor address register. Read, Write, Increment, Decrement.

SROW = Status section register. Write only.

CRSR = Current row start register. Not directly accessed.

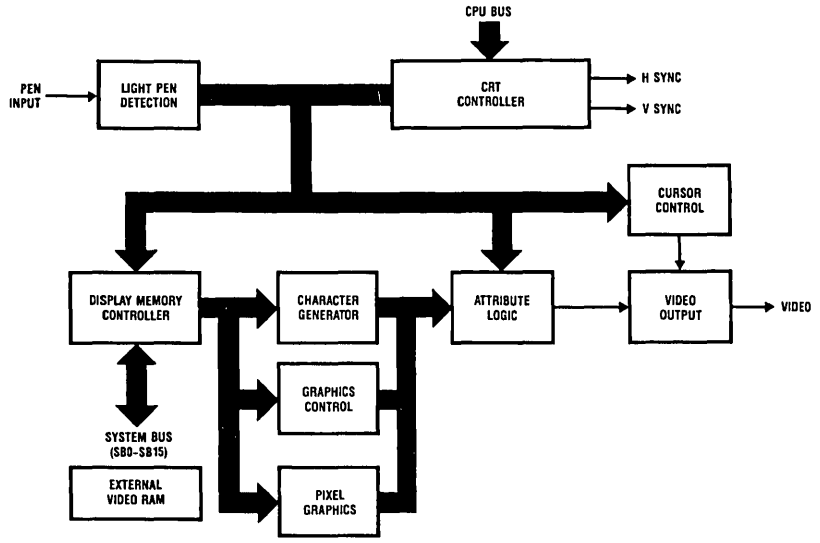
2.2.2 Sequential Access Mode

In this mode display data is accessed from sequential address locations in the video memory until the data requirements for the current screen field are fulfilled. The location from which the first display character is taken is the one pointed to by the HOME register. By modifying the contents of HOME one may implement a row scroll or paging operation. The BEGD and ENDD are used to control the wrap-around condition when HOME gets near the end of available display RAM as determined by ENDD. In this instance, when sequential accessing brings us to the end of memory as pointed to by ENDD, the controller wraps around by jumping back to the beginning of display memory as pointed to by BEGD. The value in ENDD should be the last location in display memory + 1. Also the size of the display memory between BEGD and ENDD (ENDD − BEGD) must be an integral number of display rows. The CURS in both accessing modes merely identifies the current cursor position in display memory so that the cursor characteristics can be inserted into the video at the appropriate character position.

In addition to the display of normal video data one may elect to have a special status section displayed using data from a separate section of video memory. The status section would consist of an integral number of display rows on the bottom of the screen. This feature operates by reloading the video RAM pointer with the contents of SROW when the desired row position at which to start the status section comes up. The particular row at which the status display starts is defined in the Timing Chain. Once the video RAM pointer is jumped to SROW, data accessing again proceeds sequentially from there until the data requirements for the current field are satisfied.

2.0 Functional Description (Continued)

TMP Video Section



TL/DD/5526-21

Whether a status section is used or not, upon accessing all of the data necessary to display a field, the video RAM pointer is reset to HOME in preparation for the display of a new field.

2.2.3 Table Lookup Mode

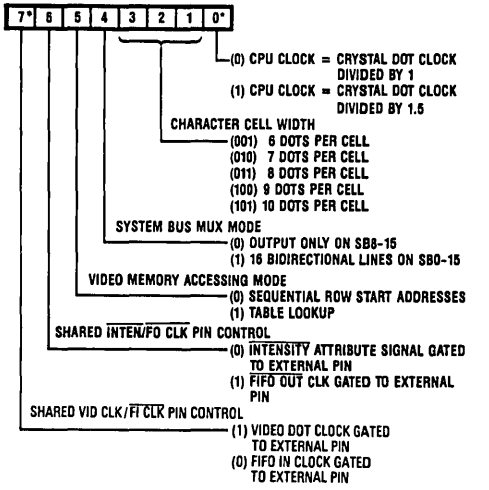
The CRSR (transparent to the user) is a pointer to the address of the first character in a display row. It is required because each time a scan line is displayed, all display characters in the row must be accessed anew. Since a row is made up of a number of scan lines, we must recover the address of the first character in the row for each scan in the row. After a row is done, the CRSR is normally advanced to point to the first character in the next row.

In table look-up mode the starting memory location of the next row is loaded into the CRSR from the HOME register at the end of each row. The HOME register was presumably updated by the CPU since the last end of row.

A CRSR load also generates the internal End of Row interrupt which the CPU will use as a signal to reload HOME. Finally, reloading HOME will clear out the End of Row interrupt. If the status section feature is used, upon reaching the begin status row location the CRSR will be loaded with SROW instead of HOME for that row. After which CRSR will revert back to load from HOME for the remaining rows on the screen.

2.3 SYSTEM CONTROL REGISTER

Through the System Control Register (SCR) the user specifies several important chip operational conditions. It is an 8-bit write only register which is loaded from the CPU accumulator.



TL/DD/5526-22

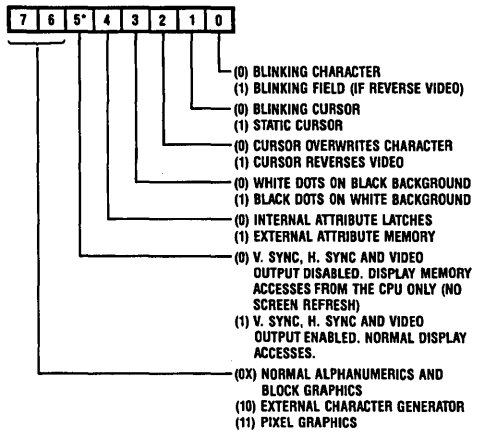
*Bit 0 is set to 1 by RESET and bit 7 is set to 0 by RESET.

FIGURE 13. System Control Register

2.4 VIDEO CONTROL REGISTER

Through the Video Control Register (VCR) the user specifies several video display features to the chip. It is an 8-bit write only register which is loaded from the CPU accumulator.

2.0 Functional Description (Continued)



TL/DD/5526-23

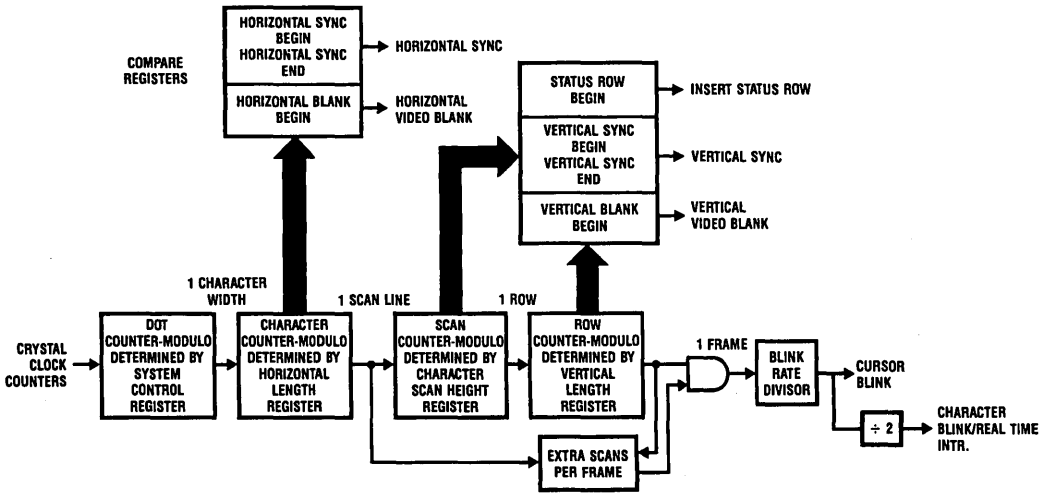
*Bit 5 is set to 0 by RESET.

FIGURE 14. Video Control Register

2.5 CRT REFRESH LOGIC

All video timing and clocking signals are derived from a series of counters and comparators called the Video Timing Chain. The chain is driven by the dot/crystal clock and ultimately divides down to the very slow blink clock, *(Figure 15)*. By having the program initialize the registers in the chain a user may specify all aspects of video generation.

The chain also controls the size and placement of the cursor and underline attribute within a character cell as well as the cell partitioning for block graphics display. All totaled, the chain consists of 14 wire only registers. They are loaded indirectly by using the Timing Chain Pointer (TCP), a 4-bit pointer to registers in the chain, and the MOV @TCP, A instruction.



TL/DD/5526-24

FIGURE 15. TMP Video Timing Chain

2.5.1 TMP Timing Chain Registers

TCP	Horizontal Timing
0	Horizontal Length Register — HLR 7 bits
	— Total number of character cells in a horizontal scan and retrace.
	— Enter desired count − 1
1	Horizontal Blank Begin Register — HBR 7 bits (Characters/Row)
	— Character position in horizontal scan after which horizontal blanking begins.
	— Enter desired number of displayed characters/row − 1.
2	Horizontal Sync Begin Register — HSBR 7 bits
	— Character position in horizontal scan after which horizontal sync begins (rising edge), HSBR ≤ HLR.
	— Enter desired count + 2.

2.0 Functional Description (Continued)

2.5.1 TMP Timing Chain Registers (Continued)

TCP — **Horizontal Timing**

3 Horizontal Sync End Register — HSER 7 bits
- — Character position in horizontal scan after which horizontal sync ends (falling edge), HSER ≤ HLR.
- — Enter desired count + 2.

Note: The polarity of the horizontal sync signal can be inverted by switching the values in the two horizontal sync registers.

TCP — **Character Height Definition**

4 High Nibble Character Scan Height Register — CSHR 4 bits (see *Figure 16a*)
- — Scan line height of a character cell.
- — Enter desired number of scan lines − 1.

4 Low Nibble Extra Scans/Frame — ES/F 4 bits
- — Number of extra scans to be added to a frame if desired.
- — Enter desired number of extra scans −1.
- — To get no extra scans make ES/F = CSHR. ES/F must be ≤ CSHR.

TCP — **Vertical Timing**

5 Vertical Length Register — VLR 5 bits
- — Total number of display and retrace rows in a frame.
- — Enter desired number of rows − 1.

6 Vertical Blank Register — VBR 5 bits (Rows/Screen)
- — Row position in vertical scan after which vertical blanking begins, VBR < VLR.
- — Enter desired number of displayed rows − 1.

7 High Nibble Vertical Sync Begin Register — VSBR 4 bits
- — Scan line position in first blank row at which vertical sync begins (falling edge). Sync starts 1 char time after blanking for that line starts (except when VSBR = CSHR sync will start 1 char time after blanking of the last displayed scan line).
- — Enter desired scan line position − 1.

7 Low Nibble Vertical Sync End Register — VSER 4 bits
- — Scan line position after start of vertical sync at which vertical sync ends (rising edge). Sync ends 1 char time after horizontal blanking for that scan line start.
- — Enter desired scan line position − 1.

Note: If VSER = VSBR there will be no vertical sync signal.

8 Status Row Begin Register — SRBR 5 bits
- — Row count after which the status row is inserted.
- — Enter desired row position − 1.

TCP — **Cursor and Graphics Control**

9 Upper 5 Bits Blink Rate 5 bits
- — Divider driven by the vertical sync frequency to yield the slow cursor, character and real-time blink rates.
- — Enter desired divisor − 1.

9 Lower 3 Bits Blink Duty Cycle 3 bits
- — Approximate ON time of blink signal.
- — 000 = shortest, 111 = longest (100 = 50% duty cycle).

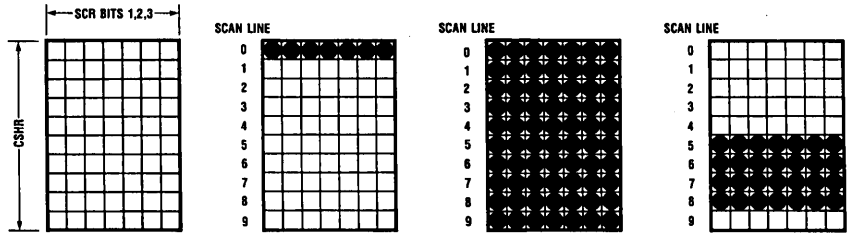
10 Graphics Column Register — GCR 8 bits
- — Assign dot positions to left, middle and right character cell columns for block graphics operation.

11 Graphics Row Register — GRR 8 bits
- — Defines scan count at which middle row for block graphics characters begins (upper nibble) and at which bottom row begins (lower nibble). The middle row (upper nibble) must be ≥ 1.
- — Enter desired scan count − 1.

12 Underline Size Register — USR 8 bits (see *Figures 16a, b, c*)
- — Defines the beginning (upper nibble) and ending (lower nibble) scan lines for the underline attribute. Values must be ≤ CSHR.

13 Cursor Size Register — CSR 8 bits (see *Figures 16a, b, c*)
- — Defines the beginning (upper nibble) and ending (lower nibble) scan lines for the cursor. Values must be ≤ CSHR.

2.0 Functional Description (Continued)



a. Character Cell Format Specification

b. Underline Size Register = 90

c. Cursor Size Register = 9A (0B, 0C, 0D, 0E, 0F) may also be used

d. Cursor Size Register = 48

TL/DD/5526–25

FIGURE 16. Underline and Cursor Register Operation

Note: The internal cursor flip-flop gets set to ON whenever a scan line corresponding to the begin cursor nibble is reached, and gets set to cursor OFF whenever a scan line corresponding to the end cursor nibble is reached. The cursor attributes are inserted whenever the character position being displayed corresponds to the one pointed to by the cursor address register. A similar situation applies for characters with the underline attribute selected. Therefore, care should be taken when setting the ES/F register and setting the cursor and underline sizes. In particular the ES/F value should not be between the upper nibble and lower nibble values of the underline size register or between the upper nibble and lower nibble values of the cursor size register. To use the cursor as a pointer without displaying it, set the lower nibble of the cursor size register to a value less than CSHR and the upper nibble to a value greater than CSHR.

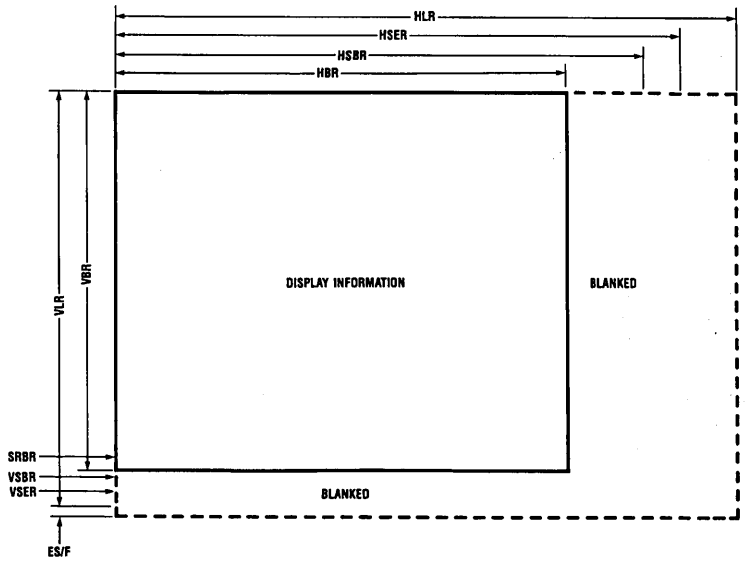
2.5.2 TIMING CHAIN LOAD VALUE EXAMPLE

It is desired to have a display field of 80 columns by 25 rows with the last screen row being a status row. It has been determined that 25 character width times will be necessary to complete horizontal retrace and that Horizontal sync should be positioned to start a full seven character times after blanking and end twenty characters after blanking to give us a total sync width of 13 character times. (See *Figure 17* for example.)

Additionally, vertical retrace will take 23 scan line times to complete with vertical sync starting three scan line times after vertical blanking begins and occupying a total period of 11 scan lines.

It is desired to make the character cells 12 scan lines tall. The cursor will be a block shape and occupy the bottom 11 scan lines in a cell. The underline attribute will actually be a strike through dash occupying the 4th scan line from the top in a cell.

Our line width is 80 displayed characters plus 25 for retrace making HLR = 80 + 25 − 1 = 104. Blanking will start after the 80th character so HBR = 80 − 1 = 79. To achieve seven character times after horizontal blanking, HSBR = 87 + 2 = 89. To achieve twenty character times after blanking HSER = 100 + 2 = 102 (note 102 − 89 = 13 total). Cell height is 12 lines so CSHR = 12 − 1 = 11. Since there are 12 scan lines per cell or row, vertical retrace will require 23/12 = 1 row and 11 scan lines. This makes our total row count VLR = 25 + 1 − 1 = 25 and ES/F = 11 − 1 = 10. Thus, timing chain location 4 would be coded: 1011 1010. We will display 25 rows so VBR = 25 − 1 = 24. Vertical sync will start at the beginning of the fourth scan



TL/DD/5526–26

FIGURE 17. Typical Video Screen Format Specification

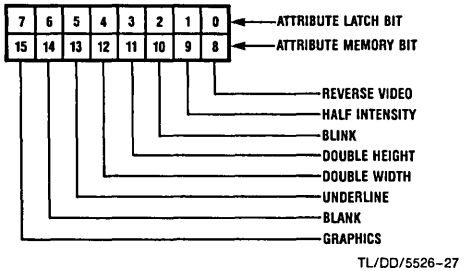
2.0 Functional Description (Continued)

line of the row after blanking begins so VSBR = 4 − 1 = 3. It will run for 11 scan lines or specifically the 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2 ending at the beginning of the 3rd so VSER = 3 − 1 = 2. The status row will be after the 24th so SRBR = 24 − 1 = 23. To specify the underline and cursor sizes one must remember that the first scan line is numbered 0. To get our 11 line block cursor we begin after the 0 line and end at the end of the 11 line making CSR = 0000 1011. The underline dash will be USR = 0011 0100. Note that the CSHR determines the scan counter modulo and if a scan compare register value (ES/F, VSBR, VSER, USR, CSR) is never reached, the signal end or begin will never be initiated.

2.6 ATTRIBUTES

Eight independent attributes may be inserted itno the video dot stream to affect display characters on either an individual or global basis. The eight attributes along with their control word bit assignments are detailed in *Figure 18*. The scope with which a particular set of attributes affects the display depends upon whether attribute control is internal or external as determined by bit 4 in the VCR.

Attributes are present if the corresponding bit is a ZERO (low).



TL/DD/5526–27

FIGURE 18. Attribute Bit Assignments

2.6.1 Internal Attribute Selection

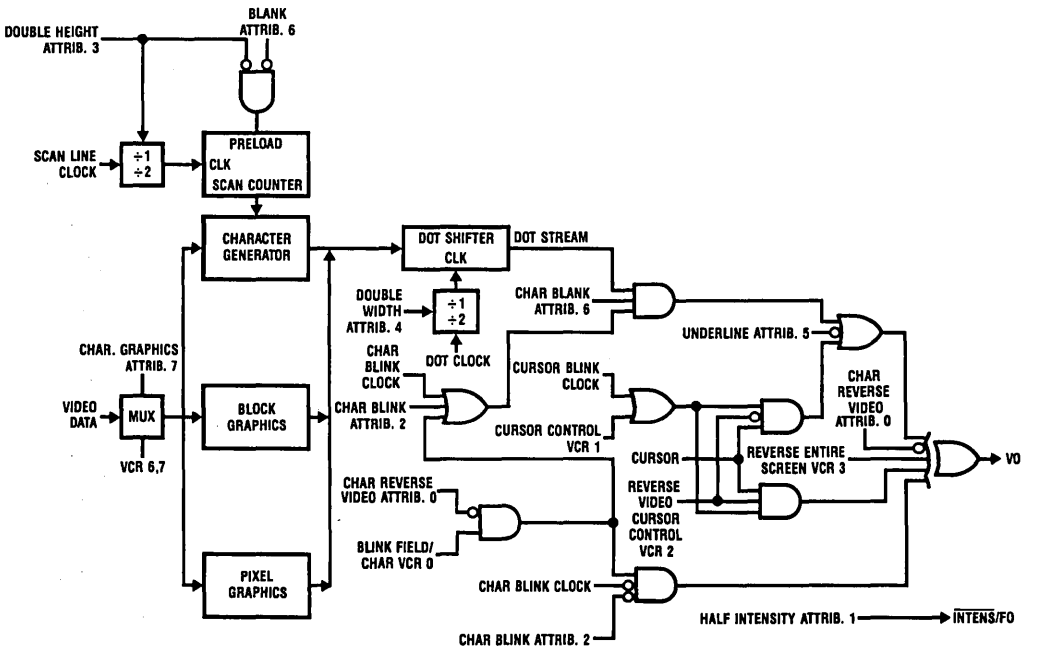
In internal mode attribute control comes from one of two internal attribute latches designated AL0 and AL1, either of which is directly loadable from the CPU accumulator. The choice of which of the two is used for a particular display character is determined by bit 7 (MSB) in the display memory data byte with 0 = AL0 and 1 = AL1. (Characters are represented in display memory as ASCII values occupying the low 7 bits of each 8-bit byte thus leaving bit 7 free for attribute control.)

2.6.2 External Attribute Selection

In external mode each display character has associated with it, a dedicated attribute field in the form of a high 8-bit extension to the regular display memory character byte. To use this mode the system bus msut be configured for 16-bit bidirectional operation (SCR bit 4 = 1) and external attributes must be selected (VCR bit 4 = 1).

2.6.3 Attribute Processing

Each of the eight attributes may be independently enabled thus yielding a number of possible combinations. The exact processing involved is shown in *Figure 19*. Note that attributes are always present. Whether any of them are active depends upon the particular control bit being enabled in the latch or memory.



TL/DD/5526–28

FIGURE 19. TMP Attribute Processing

2.0 Functional Description (Continued)

2.6.4 Attribute Operation

Reverse Video: A character and its surrounding cell are reversed in video from what was selected for the rest of the screen.

Half Intensity: To use the half intensity function the shared $\overline{\text{INTENSITY/FO CLK}}$ pin (25) must be selected for INTENSITY operation by setting SCR bit 6 low. In operation the half intensity pin will be low whenever a character for which the attribute is active is being displayed. To perform the actual attenuation function external circuitry must be connected between the $\overline{\text{INTEN}}$ and Video Output pins. In fact the signal may be used for another purpose such as switching between two colors.

Blink: A character or the field around it blinks as selected by VCR bit 0.

Double Height: A designated character is stretched out so that it will occupy a 2-row tall space. This attribute is implemented by slowing down by half the scan line stepping to the internal character generator. To use this attribute the desired double high character must be placed into the two display memory locations corresponding to the top and bottom row positions. For both locations the double high attribute is set. In addition the Blank attribute for the bottom character is also set to tell the controller it is the bottom half of a double high character. The double high attribute has no effect on element graphics or on pixel graphics displays. If an external character generator is used special circuitry must be employed to implement double high characters.

Double Width: A designated character is stretched out so that it will occupy a 2-character cell wide space. This attribute is implemented by slowing down by half the clock to the video dot shifter. To use this attribute the desired double wide character must be placed in the left character position and the double wide attribute bit set. The following character position (right) can have any character as it will be ignored.

Underline: If set this attribute causes the underline figure to be added to the video dot stream. Since the underline, like the cursor, can be specified as to position and size in the character cell, the underline can be an overline, block, strike through or any one of a number of effects. The underline overwrites any dot where it overlaps the character.

Blank/Double High Bottom: A character is inhibited from being displayed while still allowing it to be stored in the display memory. If this attribute and the double height attribute are set for the same character, the normal blank function is disabled for that character position and the character is displayed as the bottom half of a double height character.

Graphics: This attribute determines whether the video memory data byte as accessed by the display memory controller is routed through the character generator or block graphics control logic. If routed through the block graphics logic (attribute active) the effect on the video display will be as described in the Block Graphics section. Note that because Block Graphics mode is selected as an attribute it may be mixed in with normal alphanumerics characters. Also all other attributes with the exception of double height operate on the block graphics characters.

2.7 CHARACTER GENERATOR

The internal character generator holds 128 characters in a 7 x 11 matrix. The standard character sets are addressed using 7-bit ASCII codes stored in the display memory. When operating with fonts smaller than the maximum of 7 x 11, zeroes are encoded into the unused bits. When putting out a character the video controller always starts character generation on the second scan line of a row, leaving the first scan line blank. Similarly, the first (left) column in a character cell is blanked with character generation starting on the second column. Therefore, the specified cell size must be one greater in height and width than the display characters (including descenders) otherwise they will be chopped off. If the character cells are larger than the internal 7 x 11 matrix, blank dots will be put out after exhausting the internal generator (See *Figure 20* for example.)

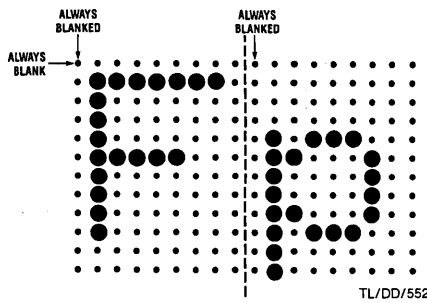


FIGURE 20. Character Cell Format

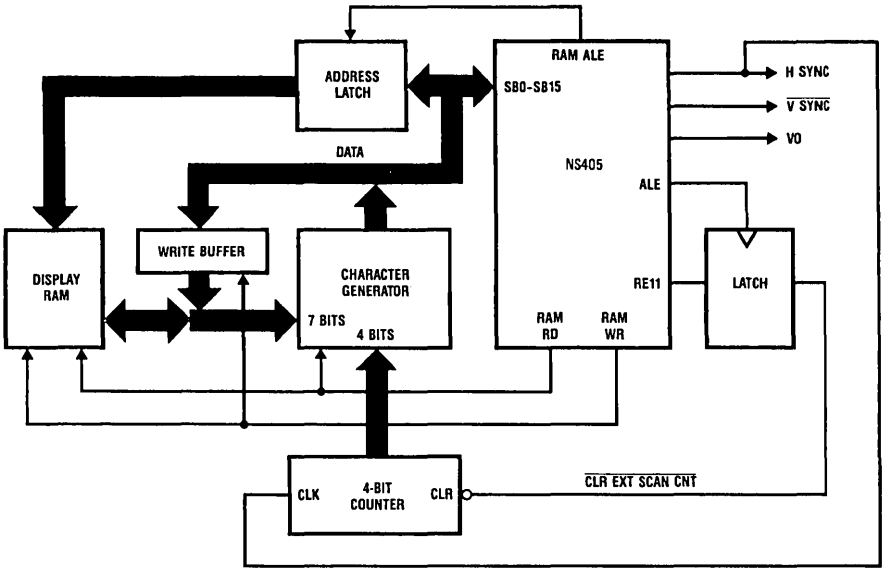
2.7.1 External Character Generation

The chip may be used with an external character generator by switching over to a pixel graphic display mode with modified address stepping as controlled by VCR bits 6, 7. In this mode an external character generator supplies pixel data to the chip as depicted in *Figure 21*. Character addressing comes from the display memory and scan line stepping from a 4-bit counter clocked by the Horizontal Sync. Scan line synchronization is achieved by using the Scan Count Clear signal coming out on RE11, pin 36. After the display of a row it pulses low to initialize the scan line counter for the start of a new row. In pixel mode both the character and any spacing between characters must be encoded into the external character generator. In addition, the chip will access and use at most 8 bits of pixel data for each character cell. However, if the cell width is specified to be 9 or 10, the ninth and tenth dots will repeat what was coded into the first. Therefore, assuming at least one dot spacing between characters, external fonts can at most be seven dots wide.

No limitations apply to the height of a character as long as the external generator can supply all of the scan lines as specified by the CSHR. As in regular pixel mode the LSB brought in is the first dot put out.

Since the eighth data bit is used for character generation it cannot effectively be used for internal attribute latch selection although one of the latches will be selected every data byte. Therefore, both internal attribute latches must be loaded with the same values. If external attribute operation is specified the full 8-bit high order attribute field is available for usage.

2.0 Functional Description (Continued)

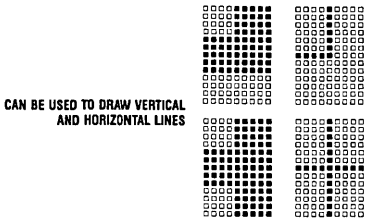


TL/DD/5526-30

FIGURE 21. External Character Set Implementation

2.8 BLOCK GRAPHICS

Block graphics is an alternative display mode to normal alphanumerics which is selected through attribute bit 7. Example *(Figure 22)*. It can operate on a character cell by character cell basis (see Attributes) and words by rerouting display memory bytes through the Block graphics logic instead of the internal character generator.



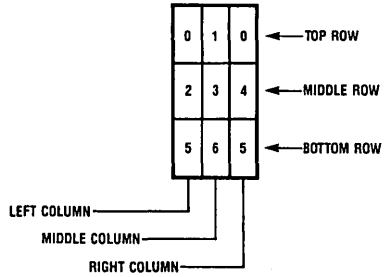
TL/DD/5526-31

FIGURE 22. Example Block Graphics Display Patterns

The Graphics Logic operates by partitioning the character cell space into nine possible areas as shown in *Figure 23* and then using the seven lower bits in the display data byte to turn these areas on or off. In this way one can draw contiguous lines or simple geometric figures while at the same time displaying alphanumeric characters in other cells.

The partitioning of the cell is controlled by two timing chain registers which specify two Horizontal and two Vertical cut off points to the graphics logic. Through these two registers one can make the sections as large or as small as desired, even eliminating sections entirely. Note that data bits 0 and 5 each control two sections as depicted in *Figure 23*.

2.8.1 Graphics Partitioning



TL/DD/5526-32

FIGURE 23. Block Graphics Cell Partitioning

The registers defining the graphics areas function as follows:

The Graphics Row Register — 8 bits (GRR) is divided into the following two (2) registers:

- Graphics Middle Row, (GMR):
  Defines the scan count at which the middle row begins (4 most significant bits of GRR).
- Graphics Bottom Row, (GBR):
  Defines the scan count at which the bottom row begins (4 least significant bits of GRR).

See *Figure 24.1a* for row example.

2.0 Functional Description (Continued)

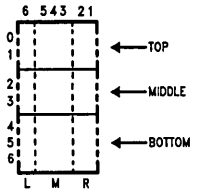
The Graphics Column Register — 8 bits (GCR) controls vertical partitioning through bit patterns as follows: (See *Figure 24.*)

7	6	5	4	3	2	1	0	
L	L	L	R	R	X	X	X	← 6 DOT FIELD WIDTH
L	L	L	R	R	R	X	X	← 7 DOT FIELD WIDTH
L	L	L	R	R	R	R	X	← 8 DOT FIELD WIDTH
L	L	L	L	R	R	R	R	← 9 DOT FIELD WIDTH

← GCR

TL/DD/5526-33

FIGURE 24. Block Graphics Column Partitioning



TL/DD/5526-44

GRR = 24
GCR = 60 (0110 0XXX)
cell size = 6 x 7

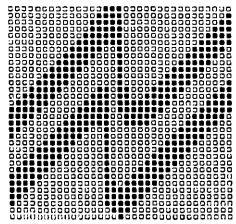
FIGURE 24.1a Block Graphics Example

For all bits in the Graphics Column Register, a one assigns that bit position to the middle column. A zero in an L bit position assigns that bit position to the left column. A zero in an R bit position assigns that bit position to the right column. There is always at least one middle dot although the left and right sections may be eliminated entirely. For 10 dot wide cells the 10th bit will repeat the 9th bit. An easy way to determine the column partitioning is to fill the GCR with all ones, thereby making it one large middle section. Then, starting from the outermost L and R bit positions, put zeros in until the left and right sections are the sizes needed.

2.9 PIXEL GRAPHICS

When bits 6 and 7 of the Video Control Register are both set to 1, the character generator and block graphics circuits are disabled. Video output directly reflects the contents of the display memory byte on a pixel (dot) per bit basis with data output LSB first. Example *(Figure 25).*

Nine bits at a time are accessed from each video memory location with as many bits being used as defined in the character cell width specification. If a cell width of 10 is specified



TL/DD/5526-34

FIGURE 25. Example Pixel Graphics

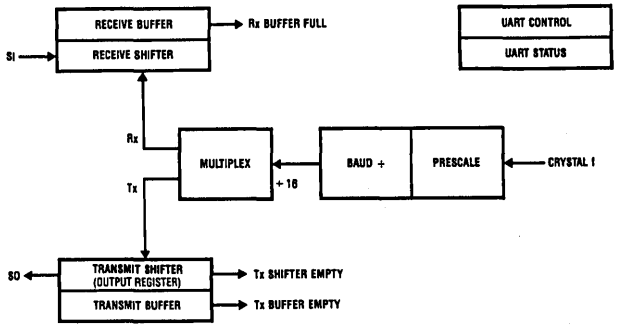
the 10th bit will merely repeat the 9th bit. Attributes are still operable in pixel mode, on a data byte basis, with internal and external operation possible. With internal attribute latch operation the same values must be loaded into both latches since the usual latch select bit is now being used for pixel control. Unless, however, only a 7 dot wide cell is used leaving the 8th bit free. With external attribute operation we are now limited to a 7-bit attribute field since pixel data can now occupy 9 of the 16 bus bits. Because of this the LSB attribute, Reverse Video is totally disabled from operation in Pixel Graphic mode. This also applies to internal attribute latch operation. Note, however, that reverse entire screen video is still operable. Address sequencing through the video memory is sequential with as many data bytes being read in as is necessary to satisfy the pixel requirements of the screen.

2.10 LIGHT PEN

Activation of the light pen interrupt causes the horizontal and vertical screen position of the currently displayed character to be latched into the Horizontal Light Pen Register HPEN (7 bits) and Vertical Light Pen Register VPEN (5 bits) respectively. Both HPEN and VPEN may be read into the CPU accumulator. The values latched remain in VPEN and HPEN until another light pen interrupt latches new values.

2.11 UART

The UART features full duplex operation with double buffered Receive and Transmit sections. Baud rate generation is fully programmable through a 2-stage divider chain. CPU control of the UART is extensive with polled or interrupt driven operation possible.



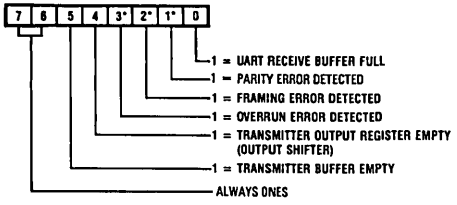
TL/DD/5526-35

FIGURE 26. TMP UART Block Diagram

2.0 Functional Description (Continued)

2.11.1 UART Control

UART Status Register (STAT): Contains error and status bits which reflect the internal state of the UART. Read into CPU accumulator. Bits 0, 5 are the same as those found in the internal interrupt register.



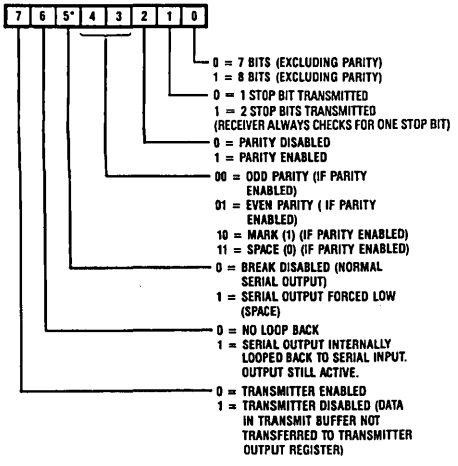
TL/DD/5526-36

UART Status Register bits 1, 2, 3 are only cleared on a chip reset or a read of the UART Receive Buffer. If another word were to come in before the Receive Buffer could be read the errors associated with the new word would add to those already present. The receipt of a new word can cause the three bits to go from a 0 to a 1, but not from a 1 to a 0.

FIGURE 27. UART Status Register

Note: The Transmit Output Register Empty flag is set to one whenever the transmitter is idle. The flag is reset to zero when a data character is transferred from the Transmit Buffer to the Output Register. This transfer does not occur until the next rising edge of the internal UART Transmit Clock. The Transmitter Output Register Empty flag occurs at the beginning of the last stop bit.

UART Control Register (UCR): Contains control bits which configure the format of transmitted data and tests made upon received data. Written to from CPU accumulator.



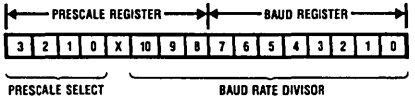
TL/DD/5526-37

*Bit 5 set to 0 by RESET.

FIGURE 28. UART Control Register

2.11.2 Baud Clock Generation

The basic BAUD clock is derived from the crystal frequency through a two-stage divider chain consisting of a 3.5–11 prescale and an 11-bit binary counter. *(Figure 29).* The divide factors are specified through 2 write only registers shown in *Figure 30.* Note that the 11-bit Baud Rate Divisor spills over into the Prescale Select Register. The correspondences between the 4-bit Prescale Select and Prescale factors is shown in Table I. There are many ways to calculate the two divisor factors but one particularly effective method would be to try to achieve a 1.8432 MHz frequency coming out of the first stage then use the BAUD Rate Divisor factors shown in Table II.



TL/DD/5526-39

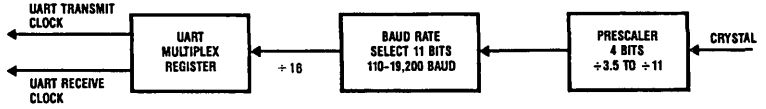
FIGURE 30. UART BAUD Clock Divider Registers

TABLE I. Prescale Factors

Prescale Select	Prescale Factor
0000	3.5
0001	4
0010	4.5
0011	5
0100	5.5
0101	6
0110	6.5
0111	7
1000	7.5
1001	8
1010	8.5
1011	9
1100	9.5
1101	10
1110	10.5
1111	11

TABLE II. Baud Rate Divisors (1.8432 MHz Input)

Baud Rate	Baud Rate Divisor (N − 1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5



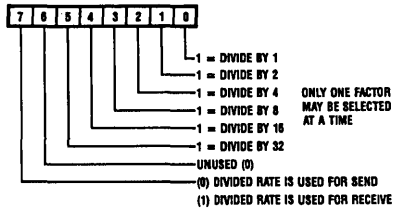
TL/DD/5526-38

FIGURE 29. UART BAUD Clock Generation

2.0 Functional Description (Continued)

The frequency coming out of the BAUD Rate Divisor is then passed through the UART Multiplex Register. Through the UART Multiplex Register one can specify that the Transmitter or Receiver clock be the same or a power of two multiple of the other.

UART Multiplex Register (UMX): Contains the bits which determine the divisor which is used to count down from the primary baud rate when different rates are used for send and receive (eight bits).



TL/DD/5526–40

FIGURE 31. UART Multiplex Register

The actual baud rate may be found from:

BR = Fc/(16*N*P*D)

Where:

BR is the Baud Rate

Fc is the external crystal frequency

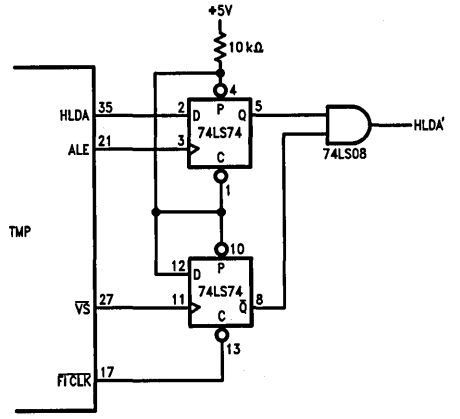
N is one plus the value of the Baud Rate Divisor contained in the Baud Rate Select Register and the Prescale Select Register.

P is the Prescale Divide Factor Selected by the value in the Prescale Select Register.

D is the Multiplex Register Divide Factor

3.0 Slave Processing

The TMP may be used as a slave video controller by having a host system perform Direct Memory Accesses into the display RAM. To assist in implementing such a system the chip features two DMA control pins—HOLD (Hold Request) and HLDA (Hold Acknowledge). These two signals come out on shared ROM Expand Bus pins RE8 and RE12. To request a DMA access a host would activate HOLD (active high and await the acknowledging HLDA from the TMP before proceeding with the DMA. The TMP only allows DMA operations during the vertical blanking period and will activate HLDA in response to a HOLD shortly after vertical blanking starts. In DMA mode all 16 TMP System Bus drivers are tri-stated while the bus control signals RAM ALE, RAM RD, RAM WR go to their inactive (high) states. A HOLD request must arrive two CPU cycles before vertical blanking starts; otherwise it will miss that retrace cycle and will have to wait until the next one, one frame later. Once DMA mode is entered, it is maintained for the duration of vertical blanking regardless of the state of HOLD. Near the end of vertical blanking the DMA mode will terminate in preparation for the display of the next frame, but the HLDA will NOT turn off. Specifically, this will occur one scan time before the end of vertical blanking. It is up to the designer to be sure that the host is off the BUS before this happens or suffer bus contention with the video controller. He can do this by either predetermining the length of time the host has to remain on the bus, or by using the end of vertical sync (as shown in *Figure 32*) to signal the end of a safe DMA period. If during DMA the CPU attempts to do a display memory access it would be put into a wait state until DMA is concluded and normal memory accessing is resumed.



TL/DD/5526–45

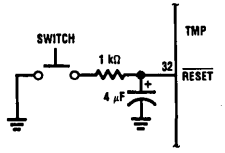
Vertical sync should be programmed to end as late as possible, but must end at least one scan time before the end of vertical blanking.

FIGURE 32

4.0 Reset

The TMP will reset if the $\overline{\text{RESET}}$ (32) pin is held at a logic low (< 0.8V) for at least five CPU cycle times. This pre-supposes that the V_{CC} is up, stable and within operational limits (+5V ±10%) and that the oscillator is running. For a power on reset, time must be allowed for the power supplies to stabilize (typically 50 ms) and the oscillator to start up. If power supply noise or ripple causes V_{CC} to exceed the +5V ±10% limits neither reset nor operation is guaranteed.

Internally, the $\overline{\text{RESET}}$ pin has a depletion load pullup that typically acts as a 30 μA current source from V_{CC} in the voltage range of interest. A typical reset circuit with a 0.5 second reset pulse is shown in *Figure 33*.



TL/DD/5526–41

FIGURE 33. Typical Reset Circuit

4.0 Reset (Continued)

During RESET a number of internal registers are initialized as follows:

4.1 CPU

CPU Clock divide = 1.5 (SCR bit 0 = 1)
Shared VIDCLK/$\overline{\text{FI CLK}}$ = 0 (SCR bit 7 = 0, $\overline{\text{FI CLK}}$ gated to external pin)
Program Counter = 0
Stack Pointer = 0
Program Memory Bank = 0
RAM Register Bank = 0
Timer Stopped
Instruction Register cleared
F0 and F1 cleared

4.2 INTERRUPTS

Internal and External Interrupts disabled
Internal Interrupt Register set to 000011X0

4.3 UART

Receiver initialized to look for start bit
Status Register set to 11110000
Transmitter initialized to wait for OUT XMTR instruction
Control Register bit 5 =0 (No BREAK)

4.4 VIDEO

Video generation shutdown (VCR bit 5 = 0)
FIFO Cleared Out
Timing Chain Character Counter = 0
Timing Chain Scan Counter = 0
Timing Chain Row Counter = 0
Timing Chain Blink Counter = 0
} IN TEST MODE ONLY

4.5 PIN STATES AT RESET

Pins 1–8 (SB0–7)	In TRI-STATE during reset and until either the CPU executes a MOVX instruction or bit 5 of the VCR is set.
Pins 9–16 (SB8–15)	If bit 4 of the SCR is set, SB8–15 will behave like SB0–7. If bit 4 of the SCR is cleared, SB8–15 will act as outputs (any of which may be either high or low). Note that bit 4 of the SCR may be one or zero at power-up.
Pin 17 (VID CLK/$\overline{\text{FI CLK}}$)	High during reset and until bit 5 of the VCR is set.
Pin 18 (RAM ALE)	High during reset and until the CPU executes a MOVX instruction or bit 5 of the VCR is set.
Pin 19 ($\overline{\text{RAM WR}}$)	High during reset and until the CPU executes a MOVX (of the output to display RAM variety) instruction.
Pin 20 ($\overline{\text{RAM RD}}$)	High during reset and until either the CPU executes a MOVX instruction or bit 5 of the VCR is set.
Pin 21 (ALE)	Pulses continuously.
Pin 22 (XTAL 2)	Crystal input or master clock input.
Pin 23 (XTAL 1)	Crystal input.
Pin 24 (Gnd.)	
Pin 25 ($\overline{\text{INTENS}}$/$\overline{\text{FO CLK}}$)	May be either high or low during reset.
Pin 26 (VO)	Low (because of asserted blanking signals) from reset until bit 5 of the VCR is set.
Pin 27 ($\overline{\text{VS}}$)	In TRI-STATE mode upon RESET, enabled when bit 5 of the VCR is set.
Pin 28 (HS)	Low from reset until bit 5 of the VCR is set.
Pin 29 (EA)	Input only. (must be tied HIGH (V_{IH2}))

4.0 Reset (Continued)

Pin 30 ($\overline{PSEN}$)	Active during reset.
Pin 31 ($\overline{RD}$)	High during reset and until an IN PORT instruction is executed.
Pin 32 ($\overline{RESET}$)	Input only.
Pin 33 (SO)	High during reset and until an OUT XMTR instruction is executed.
Pin 34 (SI)	Input only.
Pin 35 (RE12/HLDA)	If HOLD is low: low during reset. If HOLD is high: low at falling edge of ALE and during $\overline{PSEN}$, may be low or high at rising edge of ALE.
Pin 36 (RE11/$\overline{SC\ CLR}$)	If reset asserted: low at falling edge of ALE and during $\overline{PSEN}$, sampled value of internal Scan Count Clear signal is output at rising edge of ALE.
Pin 37 (RE10/$\overline{INTR}$) Pin 38 (RE9/$\overline{LPEN}$) Pin 39 (RE8/HLDR)	If reset asserted: low at falling edge of ALE and during $\overline{PSEN}$. Always in TRI-STATE at rising edge of ALE.
Pins 40–47 (RE0–7; I/O0–7)	If reset asserted: low at falling edge of ALE, in TRI-STATE during $\overline{PSEN}$, and may be either high or low at the rising edge of ALE.
Pin 48 (V_{CC})	

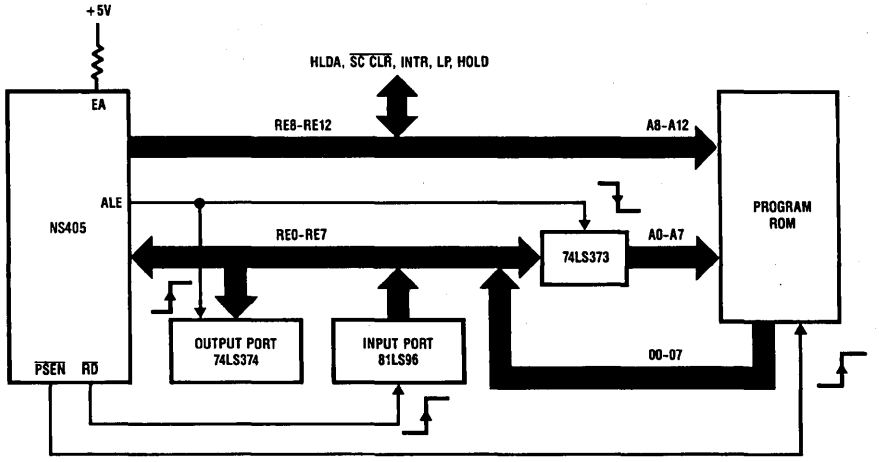
5.0 Extra Attributes

One may want to expand the external attribute field by adding more bits so that functions such as color (Red—Green—Blue drive) or grey scale may be implemented. Like the eight attributes which the chip handles internally these extra attributes would operate on a character cell basis. To add attribute bits one would have to duplicate the internal 4 level character/attribute FIFO externally using fast MSI chips. To assist in handling the external FIFO circuitry the TMP features two FIFO clocking signals on pins 17 and 25. The FIFO IN Clock ($\overline{FI\ CLK}$) is used to strobe attribute data into the external FIFO circuits in synchronism with the internal TMP FIFO. Its timing is identical to $\overline{RAM\ RD}$ but is only active when the video does a display RAM read to load its FIFO. The FIFO OUT Clock ($\overline{FO\ CLK}$) pulses for 1–3 bit times each time the video starts the display of a new character cell. The external FIFO would use the rising edge of this signal to clock out or latch the attribute output.

In order for the TMP CPU to access the additional attribute bits special bus gating arrangements would have to be worked out on the System Bus (Video Data Bus is at most 16 bits wide). Unless one were to run with internal attributes or only use a few of the external attributes in which case the unused bits could be used with the external FIFO. Whenever using the $\overline{FO\ CLK}$ the Intensity attribute is disabled since they both share the same pin.

6.0 TMP BUS Interfacing

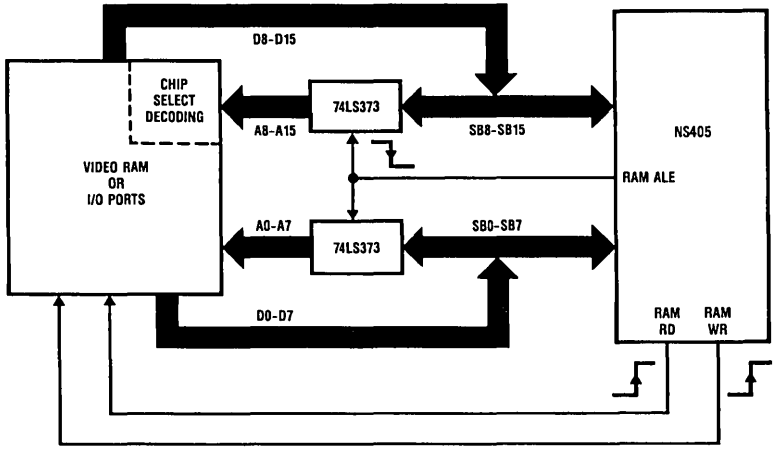
The two external buses on the TMP, ROM Expand and System are easily interfaced to as shown in *Figures 34* and *35*. Important bus information output from the chip is latched using the rising or falling edges of the various control signals. I/O port information is read in through a TRI-STATE® buffer chip such as an 81LS96.



TL/DD/5526–42

FIGURE 34. TMP ROM Expand BUS

6.0 TMP BUS Interfacing (Continued)



TL/DD/5526-43

FIGURE 35. TMP System Bus

TMP Registers (Excluding Timing Chain Registers)

TMP Registers		Associated Intructions			
		CPU SECTION			
A	= Accumulator — 8 bits	ADD A,Rr		MOV A,Rr	XCH A,Rr
#data	= data immediate	ADD A,#data		MOV A,@Rr	XCH A,@Rr
Rr	= Register	ADD A,@Rr		MOV A,#data	XCHD A,@Rr
@Rr	= Register pointed to by R0 or R1	ADDC A, Rr		MOV Rr,A	XRL A,Rr
		ADDC A,#data		MOV Rr,#data	XRL A,@Rr
		ADDC A,@Rr		MOV @Rr,A	XRL A,#data
		ANL A,Rr		MOV @Rr,#data	JBn addr
		ANL A,#data		MOVP A,@A	JNZ addr
		ANL A,@Rr		MOVP3 A,@A	JZ addr
		CLR A		RL A	DJNZ Rr,addr
		CPL A		RLC A	
		DAA		RR A	
		DEC A		RRC A	
		DEC Rr		ORL A,Rr	
		INC A		ORL A,@Rr	
		INC Rr		ORL A,#data	
		INC @Rr		SWAP A	
*HACC	= High Accumulator — 8 bits	*MOV A,HACC		*MOV HACC,A	
C	= Carry Bit	CLR C	CPL C	JNC addr	JC addr
*LONG R0	= Register Pair, R0, RA	*DECL R0		*INCL R0	*MOVL A,R0
		*MOVL R0,A		*MOVX A,@R0	*MOVX @R0,A
*LONG R1	= Register Pair R1, RB	*DECL R1		*INCL R1	*MOVL A,R1
		*MOVL R1,A		*MOVX A,@R1	*MOVX @R1,A
T	= Timer — 8 bits	MOV A,T		MOV T,A	STOP T
		STRT T		*JNTF addr	JTF addr
F0	= Flag 0	CLR F0	CPL F0	JF0 addr	*JNF0 addr
F1	= Flag 1	CLR F1	CPL F1	JF1 addr	*JNF1 addr
INTR	= Interrupt Register — 8 bits	MOV A,INTR		JNXI addr	JXI addr
		*DIS II		DIS XI	*EN II
		EN XI			

TMP Registers (Excluding Timing Chain Registers) (Continued)

TMP Registers		Associated Instructions		
CPU SECTION (Continued)				
MASK	= Internal Interrupt MasK — 8 bits	*MOV MASK,A		
PSW	= Program Status Word — 8 bits	MOV A,PSW	MOV PSW,A	
PORT	= 8 bit I/O Port	ANL PORT,#data	IN PORT	
		ORL PORT,#data	OUT PORT	
Miscellaneous Instructions		CALL addr	JMP addr	JMPP @A
		NOP	RET	RETR
		SEL MB0	SEL MB1	*SEL MB2
		*SEL MB3	SEL RB0	SEL RB1
VIDEO MANAGEMENT				
		Associated Instructions		
SCR	= System Control Register — 8 bits		*MOV SCR,A	
VCR	= Video Control Register — 8 bits		*MOV VCR,A	
HOME	= Home Address Register — 16 bits		*MOV A,HOME	*MOV HOME,A
CURS	= Cursor Address Register — 16 bits	*DEC CURS	*INC CURS	*MOVX A,@CURS
		*MOV CURS,A	*MOV A,CURS	*MOVX @CURS,A
BEGD	= Beginning of Display RAM Register — 16 bits		*MOV BEGD,A	
ENDD	= End of Display RAM Register — 16 bits		*MOV ENDD,A	
SROW	= Status Row Register — 16 bits		*MOV SROW,A	
AL0	= Attribute Latch 0 — 8 bits		*MOV AL0,A	
AL1	= Attribute Latch 1 — 8 bits		*MOV AL1,A	
HPEN	= Horizontal Light Pen Register — 7 bits		*MOV A,HPEN	
VPEN	= Vertical Light Pen Register — 5 bits		*MOV A,VPEN	
VINT	= Vertical Interrupt Register — 5 bits		*MOV VINT,A	
UART CONTROL				
PSR	= Prescale Register (UART) — 8 bits		*MOV PSR,A	
BAUD	= Baud Rate Select Register — 8 bits		*MOV BAUD,A	
UCR	= UART Control Register — 8 bits		*MOV UCR,A	
UMX	= UART Multiplex Register — 8 bits		*MOV UMX,A	
STAT	= Status Latch (UART) — 6 bits		*MOV A,STAT	
RCVR	= UART Receive Buffer — 8 bits		*IN RCVR	
XMTR	= UART Transmit Buffer — 8 bits		*OUT XMTR	
TCP	= Timing Chain Pointer		*MOV TCP,A	
@TCP	= Register Pointed to by TCP		*MOV @TCP,A	

*New instruction added to 8048 subset.

Symbol Definitions

Symbol	Definition
AC	Auxiliary Carry Flag
addr	Program Memory Address
b	Bit Designator (b = 0 – 7)
BS	RAM Bank Switch
data	Number or Expression (8 bits)
DBF	Program Memory Bank Select Bits (2)
EXI	External Interrupt Pin
F0, F1	Internal Flags
P	I/O Port (8 bits)

Symbol	Definition
PC	Program Counter
SP	Stack Pointer
TF	Timer Flag
#	Prefix for Immediate Data
@	Prefix for Indirect Address
( )	Contents of Register
(( ))	Contents of Memory Location pointed to by designated register
←	Replaced by

Instruction Set

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
ADD A, Rr	0 1 1 0 1 r r r	(A) ← (A) + (Rr) for r= 0 – 7	Add contents of designated register to the Accumulator (8-bit operation)	1	1	*	*	*		
ADD A, #data	0 0 0 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) + data	Add immediate the specified data to the Accumulator (8-bit operation)	2	2	*	*	*		
ADD A, @ Rr	0 1 1 0 0 0 0 r	(A) ← (A) + ((Rr)) for r = 0 – 1	Add indirect the contents of data memory pointed to by Rr to the Accumulator (8-bit operation)	1	1	*	*	*		
ADDC A, Rr	0 1 1 1 1 r r r	(A) ← (A) + (C) + (Rr) for r = 0 – 7	Add with carry the contents of the designated register to the Accumulator (8-bit operation)	1	1	*	*	*		
ADDC A, # data	0 0 0 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) + (C) + data	Add immediate with carry the specified data to the Accumulator (8-bit operation)	2	2	*	*	*		
ADDC A, @ Rr	0 1 1 1 0 0 0 r	(A) ← (A) + (C) + ((Rr)) for r = 0 – 1	Add indirect with carry the contents of data memory pointed to by Rr to the Accumulator (8-bit operation)	1	1	*	*	*		
ANL A, Rr	0 1 0 1 1 r r r	(A) ← (A) AND (Rr) for r = 0 – 7	Logical AND contents of designated register with Accumulator (8-bit operation)	1	1					
ANL A, # data	0 1 0 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) AND data	Logical AND specified Immediate Data with Accumulator (8-bit operation)	2	2					
ANL A, @ Rr	0 1 0 1 0 0 0 r	(A) ← (A) AND ((Rr)) for r = 0 – 1	Logical AND indirect the contents of data memory pointed to by Rr with Accumulator (8-bit operation)	1	1					
ANL PORT, # data	0 1 1 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(P) ← (P) AND data	Logical AND immediate specified data with output port (8-bit operation)	2	2					
CALL addr	a10 a9 a8 1 0 1 0 0 a7 a6 a5 a4 a3 a2 a1 a0	((SP)) ← (PC0–12) ((SP)) ← (PSW3–7) (SP) ← (SP) + 1 (PC8–10) ← addr 8–10 (PC0–7) ← addr 0–7 (PC11–12 ← DBF 0, 1	Call designated subroutine	2	2					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
CLR A	0 0 1 0 0 1 1 1	(A) ← 0	Clear the Accumulator	1	1					
CLR C	1 0 0 1 0 1 1 1	(C) ← 0	Clear carry bit	1	1	*				
CLR F0	1 0 0 0 0 1 0 1	(FO) ← 0	Clear Flag 0	1	1				*	
CLR F1	1 0 1 0 0 1 0 1	(F1) ← 0	Clear Flag 1	1	1					*
CPL A	0 0 1 1 0 1 1 1	(A) ← NOT (A)	Complement the contents of the Accumulator (8-bit operation)	1	1					
CPL C	1 0 1 0 0 1 1 1	(C) ← NOT (C)	Complement carry bit	1	1	*				
CPL F0	1 0 0 1 0 1 0 1	(F0) ← NOT (F0)	Complement Flag 0	1	1				*	
CPL F1	1 0 1 1 0 1 0 1	(F1) ← NOT (F1)	Complement Flag 1	1	1					*
DA A	0 1 0 1 0 1 1 1		Decimal Adjust the contents of the Accumulator (8-bit operation)	1	1	*	*			
DEC A	0 0 0 0 0 1 1 1	(HACC, A) ← (HACC, A) − 1	Decrement by 1 the contents of HACC/ ACC	1	1	*		*		
DEC CURS	0 0 0 0 1 0 1 0	(CURS) ← (CURS) − 1	Decrement by 1 the contents of the Cursor Address Register	1	1					
DEC Rr	1 1 0 0 1 r r r	(Rr) ← (Rr) − 1	Decrement by 1 the contents of the designated register (8-bit operation)	1	1	*				
DECL Rr	0 0 0 0 1 0 0 r	(Rr) ← (Rr) − 1 for r = 0 − 1	Decrement by 1 the contents of the designated 16-bit register pair	1	1					
DIS II	0 0 1 1 0 1 0 1		Disable internal interrupts	1	1					
DIS XI	0 0 0 1 0 1 0 1		Disable external interrupts	1	1					

DJNZ Rr, addr	1 1 1 0 1 r r r
	a7 a6 a5 a4 a3 a2 a1 a0
		(Rr) ← (Rr) − 1 for r = 0 − 7
		If (Rr) ≠ 0 do (PC0–7) ← addr
		If (Rr) = 0 do (PC) ← PC + 2
			Decrement the specified register and Jump if not zero to designated address within page (8-bit decrement)	2	2					

EN II	0 0 1 0 0 1 0 1		Enable internal interrupts.	1	1					
EN XI	0 0 0 0 0 1 0 1		Enable external interrupt.	1	1					
INC A	0 0 0 1 0 1 1 1	(HACC, A) ← (HACC, A) + 1	Increment by 1 the contents of HACC/A.	1	1	*		*		
INC CURS	0 0 1 1 1 0 1 0	(CURS) ← (CURS) + 1	Increment by 1 the contents of the Cursor Address Register.	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
INC Rr	0 0 0 1 1 r r r	(Rr) ← (Rr) + 1 for r = 0 − 7	Increment by 1 the contents of the designated register (8-bit increment)	1	1	*				
INC @ Rr	0 0 0 1 0 0 0 r	((Rr)) ← ((Rr)) + 1 for r = 0 − 1	Increment in direct the contents of data memory pointed to by Rr (8-bit increment)	1	1	*				
INCL Rr	0 0 1 1 1 0 0 r	(Rr) ← (Rr) + 1 for r = 0 − 1	Increment by 1 the contents of the designated 16-bit register pair	1	1					
IN PORT	1 1 1 0 0 0 0 1	(A) ← (P)	Input data from port into Accumulator (8-bit transfer)	2	1					
IN RCVR	1 1 1 0 0 0 0 0	(A) ← (RCVR)	Input contents of UART Receive buffer into Accumulator (8-bit transfer). Also, clears Receive Buffer Full interrupt.	1	1					
JBb addr	b2 b1 b0 1 0 0 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0-7) ← addr if (b) = 1 (PC) ← (PC) + 2 if (b) = 0 for b = 0 − 7	Jump to specified address within page if Accumulator bit is set	2	2					
JC addr	1 1 1 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0-7) ← addr if C = 1 (PC) ← (PC) + 2 if C = 0	Jump to specified address within page if Carry flag is set	2	2					
JF0 addr	1 0 0 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0-7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address within page if Flag F0 is set	2	2					
JF1 addr	0 1 1 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0-7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address within page if Flag F1 is set	2	2					
JMP addr	a10 a9 a8 0 0 1 0 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC8-10) ← addr 8-10 (PC0-7) ← addr 0-7 (PC11-12) ← DBF 0, 1	Direct Jump to specified address within 2k Bank	2	2					
JMPP @ A	1 0 1 0 0 0 1 1	(PC0-7) ← ((A))	Jump indirect within page to the address specified in the memory location pointed to by the Accumulator	2	1					
JNC addr	1 1 1 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0-7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump within page to specified address if Carry flag is 0	2	2					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
JNF0 addr	1 0 0 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if F0 = 0 (PC) ← (PC) + 2 if F0 = 1	Jump within page to specified address if F0 is 0	2	2					
JNF1 addr	0 1 1 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if F1 = 0 (PC) ← (PC) + 2 if F1 = 1	Jump within page to specified address if F1 is 0	2	2					
JNTF addr	0 0 0 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if TF = 0 (PC) ← (PC) + 2 if TF = 1, (TF) ← 0	Jump within page to specified address if Timer flag is reset. If not, continue and reset TF	2	2					
JNXI addr	1 0 1 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if EXI = LOW (PC) ← (PC) + 2 if EXI = HIGH	Jump within page to specified address if External Interrupt pin is LOW	2	2					
JNZ addr	1 1 0 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if A ≠ 0 (PC) ← (PC) + 2 if A = 0	Jump within page to specified address if Accumulator is not 0	2	2					
JTF addr	0 0 0 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if TF = 1, (TF) ← 0 (PC) ← (PC) + 2 if TF = 0	Jump within page to specified address if Timer flag is set. If jump taken Timer flag reset	2	2					
JXI addr	1 0 1 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if EXI = HIGH (PC) ← (PC) + 2 if EXI = LOW	Jump within page to specified address if External Interrupt pin is HIGH	2	2					
JZ addr	1 1 0 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if A = 0 (PC) ← (PC) + 2 if A ≠ 0	Jump within page to specified address if Accumulator is 0	2	2					
MOV A, CURS	1 0 0 1 1 0 1 1	(HACC/A) ← (CURS)	Copy the contents of the Cursor Address Register into the HACC/A (16-bit transfer)	1	1			*		
MOV A, HACC	1 1 1 0 0 0 1 0	(A) ← (HACC)	Copy contents of the High Accumulator into the Low Accumulator (8-bit transfer)	1	1					
MOV A, HOME	1 0 0 1 1 0 1 0	(HACC/A) ← (HOME)	Copy the contents of the Home Address register into the HACC/A (16-bit transfer)	1	1			*		
MOV A, HPEN	0 0 1 1 1 1 1 1	(A0–6) ← (HPEN) (A7) ← 0	Copy the contents of the Horizontal Light Pen Register into the Accumulator (7-bit transfer, A7 cleared)	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOV A, INTR	1 0 0 0 1 1 0 0	(A) ← (INTR)	Copy the contents of the Interrupt Register into the Accumulator (8-bit transfer)	1	1					
MOV A, PSW	1 1 0 0 0 1 1 1	(A) ← (PSW)	Copy contents of the Program Status word into the Accumulator (8-bit transfer)	1	1					
MOV A, Rr	1 1 1 1 1 r r r	(A) ← (Rr) for r = 0 – 7	Copy the contents of the designated Register into the Accumulator (8-bit transfer)							
MOV A, STAT	1 0 0 1 1 1 0 0	(A0–5) ← (STAT) (A6–7) ← 11	Copy the contents of the UART Status Latch into the Accumulator (6-bit transfer, A6 and A7 set)	1	1					
MOV A, T	0 1 0 0 0 0 1 0	(A) ← (T)	Copy the contents of the Timer into the Accumulator (8-bit transfer)	1	1					
MOV A, VPEN	0 0 1 1 1 1 1 0	(A0–4) ← (VPEN) (A5–7) ← 0	Copy contents of the Vertical Light Pen Register into the Accumulator (5-bit transfer, A5–A7 cleared)	1	1					
MOV A, @ Rr	1 1 1 1 0 0 0 r	(A) ← ((Rr)) for r = 0 – 1	Copy indirect the contents of data memory pointed to by Rr into the Accumulator (8-bit transfer)	1	1					
MOV A, # data	0 0 1 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← data	Load immediate the specified data into the Accumulator (8-bit load)	2	2					
MOV AL0, A	0 0 1 1 1 1 0 0	(AL0) ← (A)	Copy the contents of the Accumulator into Attribute Latch 0 (8-bit transfer)	1	1					
MOV AL1, A	0 0 1 1 1 1 0 1	(AL1) ← (A)	Copy the contents of the Accumulator into Attribute Latch 1 (8-bit transfer)	1	1					
MOV BAUD, A	0 0 0 0 0 0 1 0	(BAUD) ← (A)	Copy the contents of the Accumulator into the UART Baud Rate Select Register (8-bit transfer)	1	1					
MOV BEGD, A	0 0 0 0 1 1 0 1	(BEGD) ← (HACC/A)	Copy the contents of HACC/A into the Beginning of Display RAM Register (16-bit transfer)	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOV CURS, A	1 0 0 0 1 0 1 1	(CURS) ← (HACC/A)	Copy the contents of HACC/A into the Cursor Address Register (16-bit transfer)	1	1					
MOV ENDD, A	0 0 0 0 1 1 0 0	(ENDD) ← (HACC/A)	Copy the contents of HACC/A into the End of Display RAM Register (16-bit transfer)	1	1					
MOV HACC, A	1 1 0 0 0 0 1 0	(HACC) ← (A)	Copy the contents of the Low Accumulator into the High Accumulator (8-bit transfer)	1	1			*		
MOV HOME, A	1 0 0 0 1 0 1 0	(HOME) ← (HACC/A)	Copy the contents of HACC/A into the Home Address Register (16-bit transfer)	1	1					
MOV MASK, A	1 0 0 0 0 0 1 0	(MASK) ← (A)	Copy the contents of the Accumulator into the Interrupt Mask Register (8-bit transfer)	1	1					
MOV PSR, A	0 0 1 0 0 0 1 0	(PSR) ← (A)	Copy the contents of the Accumulator into the UART Prescale Register (8-bit transfer)	1	1					
MOV PSW, A	1 1 0 1 0 1 1 1	(PSW) ← (A)	Copy contents of the Accumulator into the Program Status Word (8-bit transfer)	1	1	*	*			
MOV Rr, A	1 0 1 0 1 r r r	(Rr) ← (A) for r = 0 – 7	Copy contents of the Accumulator into the designated register (8-bit transfer)	1	1					
MOV SCR, A	0 1 0 1 0 1 0 1	(SCR) ← (A)	Copy contents of the Accumulator into the System Control Register (8-bit transfer)	1	1					
MOV SROW, A	0 0 0 0 1 1 1 0	(SROW) ← (HACC/A)	Copy the contents of HACC/A into the Status Row Register (16-bit transfer)	1	1					
MOV T, A	0 1 1 0 0 0 1 0	(T) ← (A)	Copy the contents of the Accumulator into the Timer (8-bit transfer)	1	1					
MOV TCP, A	1 0 0 0 0 1 1 1	(TCP) ← (A)	Copy the contents of the Accumulator into the Timing Chain Pointer	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOV UCR, A	0 0 0 0 0 0 0 1	(UCR) ← (A)	Copy the contents of the Accumulator into the UART Control Register (8-bit transfer)	1	1					
MOV VCR, A	0 1 0 0 0 1 0 1	(VCR) ← (A)	Copy the contents of the Accumulator into the Video Control Register (8-bit transfer)	1	1					
MOV VINT, A	1 0 1 0 0 0 1 0	(VINT) ← (A)	Copy the contents of the Accumulator into the Vertical Interrupt Register	1	1					
MOV Rr, # data	1 0 1 1 1 r r r d7 d6 d5 d4 d3 d2 d1 d0	(Rr) ← data for r = 0 − 7	Load immediate the specified data into the designated register (8-bit load)	2	2					
MOV @ Rr, A	1 0 1 0 0 0 0 r	((Rr)) ← (A) for r = 0 − 1	Copy indirect the contents of the Accumulator into the data memory location pointed to by Rr (8-bit transfer)	1	1					
MOV @ Rr, # data	1 0 1 1 0 0 0 r d7 d6 d5 d4 d3 d2 d1 d0	((Rr)) ← data for r = 0 − 1	Load indirect the specified immediate data into the data memory location pointed to by Rr (8-bit load)	2	2					
MOV @ TCP, A	1 0 1 1 0 1 1 1	((TCP)) ← (A) (TCP) ← (TCP) + 1	Copy indirect the contents of the Accumulator into the Timing Chain Register pointed to by TCP. Contents of TCP incremented by 1	1	1					
MOV UMX, A	0 0 1 1 0 0 1 1	(UMX) ← (A)	Copy the contents of the Accumulator into the UART Multiplex Register (8-bit transfer)	1	1					
MOVL A, R0	1 0 0 1 1 0 0 0	(HACC/A) ← (RA, R0)	Copy the contents of RA, R0 into HACC/A (16-bit transfer)	1	1			*		
MOVL A, R1	1 0 0 1 1 0 0 1	(HACC/A) ← (RB, R1)	Copy the contents of RB, R1 into HACC/A (16-bit transfer)	1	1			*		
MOVL R0, A	1 0 0 0 1 0 0 0	(RA, R0) ← (HACC/A)	Copy the contents of HACC/A into RA, R0 (16-bit transfer)	1	1					
MOVL R1, A	1 0 0 0 1 0 0 1	(RB, R1) ← (HACC/A)	Copy the contents of HACC/A into RB, R1 (16-bit transfer)	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOVP A, @ A	1 0 1 1 0 0 1 1	(PC0–7) ← (A) (A) ← ((PC)) (PC0–7) ← (old PC0–7) + 1	Replace low 8 bits of PC with A. Load indirect within page the contents of the memory location pointed to by new PC into Accumulator. Restore PC with old value plus 1. Operates in all memory banks.	2	1					
MOVP3 A, @ A	1 1 1 1 0 0 1 1	(PC0–7) ← (A) (PC8–10) ← 011 (A) ← ((PC)) (PC) ← (old PC) + 1	Replace low 8 bits of PC with A. Next 3 bits replaced with 011. Load indirect within page 3 the contents of the memory location pointed to by new PC into the Accumulator. Restore PC with old value plus 1. Operates in all memory banks.	2	1					
MOVX A, @ CURS	1 0 0 1 1 1 0 1	(HACC/A) ← ((CURS))	Copy indirect the contents of display memory as pointed to by CURS into HACC/A (16-bit transfer)	Min. 2	1			*		
MOVX A, @ R0	1 0 0 1 0 0 0 0	(HACC/A) ← ((RA, R0))	Copy indirect the contents of display memory as pointed to by RA, R0 into HACC/A (16-bit transfer)	Min. 2	1			*		
MOVX A, @ R1	1 0 0 1 0 0 0 1	(HACC/A) ← ((RB, R1))	Copy indirect the contents of display memory as pointed to by RB, R1 into HACC/A (16-bit transfer)	Min. 2	1			*		
MOVX @ CURS, A	1 0 0 0 1 1 0 1	((CURS)) ← (HACC/A)	Copy indirect the contents of HACC/A into the display memory location as pointed to by CURS (16-bit transfer)	Min. 2	1					
MOVX @ R0, A	1 0 0 0 0 0 0 0	((RA, R0)) ← (HACC/A)	Copy indirect the contents of HACC/A into the display memory location as pointed to by RA, R0 (16-bit transfer)	Min. 2	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOVX @ R1, A	1 0 0 0 0 0 0 1	((RB, R1)) ← (HACC/A)	Copy indirect the contents of HACC/A into the display memory location pointed to by RB, R1 (16-bit transfer)	Min. 2	1					
NOP	0 0 0 0 0 0 0 0		No Operation	1	1					
ORL A, Rr	0 1 0 0 1 r r r	(A) ← (A) OR (Rr) for r = 0 – 7	Logical OR contents of designated register with Accumulator (8-bit transfer)	1	1					
ORL A, @ Rr	0 1 0 0 0 0 0 r	(A) ← (A) OR ((Rr)) for r = 0 – 1	Logical OR indirect the contents of the data memory location pointed to by Rr with Accumulator (8-bit operation)	1	1					
ORL A, # data	0 1 0 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) OR data	Logical OR the specified immediate data with the Accumulator (8-bit operation)	2	2					
ORL PORT, # data	0 1 1 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(P) ← (P) OR data	Logical OR immediate specified data with output port	2	2					
OUT PORT	1 1 0 0 0 0 0 1	(P) ← (A)	Output the contents of the Accumulator to the I/O Port (8-bit transfer)	2	1					
OUT XMTR	1 1 0 0 0 0 0 0	(XMTR) ← (A)	Copy the contents of the Accumulator into the UART Transmit Buffer (8-bit transfer). Also clears Transmit Buffer empty interrupt	1	1					
RET	1 0 0 0 0 0 1 1	(SP) ← (SP) – 1 (PC0–12) ← ((SP))	Return from subroutine without restoring Program Status Word bits 5–7	2	1					
RETR	1 0 0 1 0 0 1 1	(SP) ← (SP) – 1 (PC0–12) ← ((SP)) (PSW 3–7) ← ((SP))	Return from Subroutine restoring Program Status Word (use for all returns from interrupts)	2	1	*	*			
RL A	1 1 1 0 0 1 1 1	(A_{n+1}) ← (An) for n = 0 – 6 (A0) ← (A7)	Rotate Accumulator left by 1 bit without carry	1	1					
RLC A	1 1 1 1 0 1 1 1	(A_{n+1}) ← (An) for n = 0 – 6 (A0) ← (C) (C) ← (A7)	Rotate Accumulator left by 1 bit through carry	1	1	*				

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
RR A	0 1 1 1 0 1 1 1	(An) ← A_{n+1} for n = 0 − 6	Rotate Accumulator right by 1 bit without carry	1	1					
RRC A	0 1 1 0 0 1 1 1	(An) ← A_{n+1} for n = 0 −6 (A7) ← (C) (C) ← (A0)	Rotate Accumulator right by 1 bit through carry	1	1	*				
SEL MB0	1 1 0 0 0 1 0 1	(DBF) ← 00	Select Bank 0 (0−2047) of Program Memory	1	1					
SEL MB1	1 1 0 1 0 1 0 1	(DBF) ← 01	Select Bank 1 (2048−4095) of Program Memory	1	1					
SEL MB2	1 1 1 0 0 1 0 1	(DBF) ← 10	Select Bank 2 (4096−6143) of Program Memory	1	1					
SEL MB3	1 1 1 1 0 1 0 1	(DBF) ← 11	Select Bank 3 (6144−8191) of Program Memory	1	1					
SEL RBn	1 1 n 0 0 0 1 1	(BS) ← n for n = 0 − 1	Select Data RAM Bank (0−7) or 1 (24−31)	1	1					
STOP T	0 1 1 0 0 1 0 1		Stop Timer	1	1					
STRT T	0 1 1 1 0 1 0 1		Start Timer	1	1					
SWAP A	0 1 0 0 0 1 1 1	(A4−A7) ⟷ (A0−A3)	SWAP 4 bit nibbles in Accumulator	1	1					
XCH A, Rr	0 0 1 0 1 r r r	(A) ⟷ (Rr) for r = 0 − 7	Exchange the Accumulator and contents of designated register (8-bit transfer)	1	1					
XCH A, @ Rr	0 0 1 0 0 0 0 r	(A) ⟷ ((Rr)) for r = 0 − 1	Exchange indirect the contents of the Accumulator and the data memory location pointed to by Rr (8-bit transfer)	1	1					
XCHD A, @ Rr	0 0 1 1 0 0 0 r	(A0−3) ⟷ ((Rr)) 0−3 for r = 0 − 1	Exchange indirect the low 4 bits of the Accumulator and the data memory location pointed to by Rr (4-bit transfer)	1	1					
XRL A, Rr	1 1 0 1 1 r r r	(A) ← (A) XOR (Rr) for r = 0 − 7	Logical XOR contents of designated register with Accumulator (8-bit transfer)	1	1					
XRL A, @ Rr	1 1 0 1 0 0 0 r	(A) ← (A) XOR ((Rr)) for r = 0 − 1	Logical XOR indirect the contents of the data memory location pointed to by Rr with the Accumulator	1	1					
XRL A, # data	1 1 0 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) XOR data	Logical XOR the immediate specified data with the Accumulator	2	2					

TMP Opcode Chart

MSN \ LSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	MOV UCR, A	MOV BAUD, A	ADD A, #data	JMP (page 0)	EN XI	JNTF	DEC A	DECL R0	DECL R1	DEC CURS		MOV ENDD, A	MOV BECD, A	MOV SROW, A	
1	INC @R0	INC @R1	JB0	ADDC A, #data	CALL (page 0)	DIS XI	JTF	INC A	INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7
2	XCH A, @R0	XCH A, @R1	MOV PSR, A	MOV A, #data	JMP (page 1)	EN II		CLR A	XCH A, R0	XCH A, R1	XCH A, R2	XCH A, R3	XCH A, R4	XCH A, R5	XCH A, R6	XCH A, R7
3	XCHD A, @R0	XCHD A, @R1	JB1	MOV UMX, A	CALL (page 1)	DIS II		CPL A	INCL R0	INCL R1	INC CURS		MOV AL0, A	MOV AL1, A	MOV A, VPEN	MOV A, HFEN
4	ORL A, @R0	ORL A, @R1	MOV A,T	ORL A, #data	JMP (page 2)	MOV VCR, A		SWAP A	ORL A, R0	ORL A, R1	ORL A, R2	ORL A, R3	ORL A, R4	ORL A, R5	ORL A, R6	ORL A, R7
5	ANL A, @R0	ANL A, @R1	JB2	ANL A, #data	CALL (page 2)	MOV SCR, A		DA A	ANL A, R0	ANL A, R1	ANL A, R2	ANL A, R3	ANL A, R4	ANL A, R5	ANL A, R6	ANL A, R7
6	ADD A, @R0	ADD A, @R1	MOV T,A	ORL PORT, #data	JMP (page 3)	STOP T	JNF1	RRC A	ADD A, R0	ADD A, R1	ADD A, R2	ADD A, R3	ADD A, R4	ADD A, R5	ADD A, R6	ADD A, R7
7	ADDC A, @R0	ADDC A, @R1	JB3	ANL PORT, #data	CALL (page 3)	STRT T	JF1	RR A	ADDC A, R0	ADDC A, R1	ADDC A, R2	ADDC A, R3	ADDC A, R4	ADDC A, R5	ADDC A, R6	ADDC A, R7
8	MOVX @R0, A	MOVX @R1, A	MOV MASK, A	RET	JMP (page 4)	CLR F0	JNF0	MOV TCP, A	MOVL R0, A	MOVL R1, A	MOV HOME, A	MOV CURS, A	MOV A, INTR	MOVX @CURS, A		
9	MOVX A, @R0	MOVX A, @R1	JB4	RETR	CALL (page 4)	CPL F0	JF0	CLR C	MOVL A, R2	MOVL A, R1	MOV A, HOME	MOV A, CURS	MOV A, STAT	MOVX A, @ CURS		
A	MOV @R0, A	MOV @R1, A	MOV VINT, A	JMPP @A	JMP (page 5)	CLR F1	JNXI	CPL C	MOV R0, A	MOV R1, A	MOV R2, A	MOV R3, A	MOV R4, A	MOV R5, A	MOV R6, A	MOV R7, A
B	MOV @R0, #data	MOV @R1, #data	JB5	MOVP A, @A	CALL (page 5)	CPL F1	JXI	MOV @TCP, A	MOV R0, #data	MOV R1, #data	MOV R2, #data	MOV R3, #data	MOV R4, #data	MOV R5, #data	MOV R6, #data	MOV R7, #data
C	OUT XMTR	OUT PORT	MOV HACC, A	SEL RB0	JMP (page 6)	SEL MB0	JZ	MOV A, PSW	DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7
D	XRL A, @R0	XRL A, @R1	JB6	XRL A, #data	CALL (page 6)	SEL MB1	JNZ	MOV PSW, A	XRL A, R0	XRL A, R1	XRL A, R2	XRL A, R3	XRL A, R4	XRL A, R5	XRL A, R6	XRL A, R7
E	IN RCVR	IN PORT	MOV A, HACC	SEL RB1	JMP (page 7)	SEL MB2	JNC	RL A	DJNZ R0	DJNZ R1	DJNZ R2	DJNZ R3	DJNZ R4	DJNZ R5	DJNZ R6	DJNZ R7
F	MOV A, @R0	MOV A, @R1	JB7	MOVP3 A, @A	CALL (page 7)	SEL MB3	JC	RLC A	MOV A, R0	MOV A, R1	MOV A, R2	MOV A, R3	MOV A, R4	MOV A, R5	MOV A, R6	MOV A, R7

Ordering Information

ORDER PART NUMBERS

ROMless	NS405-A12N NS405-B12N NS405-C12N	NS405-B18N

Throughput Considerations In NS405 System Planning

National Semiconductor
Application Brief 14
James Murashige

The intricate timing relationships inherent in video generation require that a designer have a firm grasp of the fundamentals of NS405 operation in order to achieve his design objectives. Towards this end the key facets of NS405 operation will be examined and examples given.

The NS405 is a complete video controller that reads in video data, processes it and outputs it to a CRT. Given this, one may derive all essential operating parameters from the following two statements:

1. You must be able to read in video data faster than you output it.
2. Video data accesses are based on the CPU cycle which in turn is based on the crystal or dot clock.

Application of these two statements immediately leads to a limitation on the character cell width as follows:

if f = crystal frequency or dot clock

then $(f \div 1) \div 15$ or $(f \div 1.5) \div 15$ = CPU Instruction Execution Clock Frequency

Since there are three video data accesses each CPU Instruction Execution cycle, there are $3 * (f \div 1) \div 15$ or $3 * (f \div 1.5) \div 15$ video data accesses per second.

if w = dot width of character cell then $f \div w$ = number of character cells being displayed per second.

Statement 1 says that video data accesses/sec ≥ display characters/sec

for CPU Clock ÷ 1	for CPU Clock ÷ 1.5
$3 * (f \div 1) \div 15 \geq f \div w$	$3 * (f \div 1.5) \div 15 \geq f \div w$
$f \div 5 \geq f \div w$	$(3 * f) \div 22.5 \geq f \div w$
$1/5 \geq 1/w$	$3/22.5 \geq 1/w$
$w \geq 5$	$w \geq 7.5$

So depending on the CPU clock divide factor (÷1 or ÷1.5) the character cell width must be a minimum as shown.

Cell width also impacts CPU throughput since both the CPU and Video controller vie for video memory access through the DMA controller. The rules of access are simple and straightforward. The Video Controller gets as many of the accesses as it needs with the CPU getting any left over. The maximum access rate as already shown is $f \div 5$ or $f \div 7.5$ depending on the CPU clock divide. If the CPU attempts a video memory access when things are very busy it will be put into a wait state and remain frozen until things clear up. Of course, no display characters are necessary when the display is blanked, so during the horizontal and vertical retrace periods the CPU has unlimited access to video memory.

Normally, the CPU doesn't have to wait until horizontal retrace to get into video memory, but exactly how often it can get in during a display line requires analysis of the worst case video requirements.

Since the results can vary dramatically depending on the parameters chosen, two typical cases will be presented.

I. With a dot clock of 18 MHz the display line consists of 80 character cells, 9 dots across. Since the CPU clock divide must be 1.5 the video memory access rate is 18 MHz ÷ 7.5 = 2.4 MHz.

   To display one line requires (9×80)/18 MHz = 40 us.

   In one line time there are 2.4 MHz × 40 us = 96 video memory accesses. Of the 96, 80 are required for the characters displayed in the line leaving 16 available for the CPU. This is an average of one every six video memory accesses or once every two CPU instruction cycles. This would be fine since all CPU video memory instructions require two instruction cycles to execute anyway. However, in addition to the DMA controller the video circuits also employ a four level FIFO to insure a smooth data flow. The FIFO is normally kept full at four in which case it stops accessing video data and allows the CPU to have all the accesses. However, the FIFO can drop down quite far before starting to fill up again by taking all of the video memory accesses. The net effect is that instead of being evenly distributed, the accesses available to the CPU are clumped together with long gaps between clumps. Taking the worst case condition of the FIFO being completely empty and having to fill to four by taking the accesses which the CPU could have gotten, the longest gap is $(4 \times 6) + 5 = 29$ accesses ≈ 10 CPU instruction cycles. Generally speaking this tends to happen towards the middle of a line since the FIFO is filled prior to the start of a line and tries to end a line empty. In fact, accesses for video are performed up to the second to the last display character. The FIFO prefetch for the next line is performed shortly after horizontal blanking starts.

II. If the dot clock is now 12 MHz with a display line of 80 character cells 7 dots across the CPU clock divide can be 1.

   The video memory access rate is 12 MHz ÷ 5 = 2.4 MHz.

   To do one line requires (7×80)/12 MHz = 46.7 us. In one line time there are 2.4 MHz × 46.7 us = 112 video memory accesses. Of the 112, 32 are now available to the CPU. This averages out to one every 3.5. Figuring the FIFO in, the worst case wait for the CPU becomes $(4 \times 3.5) + 2.5 = 16.5$ accesses ≈ 6 CPU instruction cycles. A significant improvement over the first example.

In general, to maximize CPU access to video memory one must maximize the average number of "free" accesses during the display time. The number of free accesses as a fraction of the total number available is:

$(w - 5d)/w$ Where w = character cell dot width
d = CPU divide factor of 1 or 1.5

As can be seen, throughput performance depends entirely on the cell width and CPU clock divide. To maximize performance one would try to choose a large w and a d of 1.

Applying the delay imposed by the four level FIFO, the maximum CPU delay in accessing video memory becomes =

$(4w + 5d)/(w - 5d)$ Memory cycles

NS405-Series TMP External Interrupt Processing

National Semiconductor
Application Brief 16
James Murashige

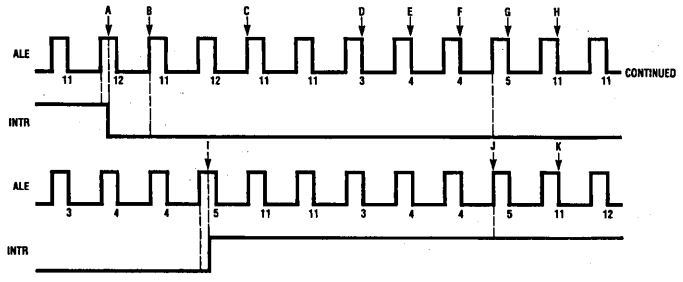
The TMP External Interrupt (INTR) is a level sampled interrupt input. Specifically this means that the input is sampled once each CPU cycle with interrupts being generated as long as the sampled input is a logic low. INTR shares pin 37 with RE10 and is sampled on each ALE rising edge as shown in the data sheet. If a logic low level is detected, interrupt service will commence if interrupts had been previously enabled with an EN XI instruction. Service consists of finishing up the currently executing instruction, pushing the PC and other pertinent information onto the stack, disabling all interrupts while in service and finally performing a JUMP to location 003. Upon completion of service a RETR would be executed to pop the stack and return to where we left off in the main program.

The exact timing involved may be observed through the example program of *Figure 1* and its instruction execution sequence in *Figure 2.* In *Figure 2* the numbers shown on the falling ALE edges are the program addresses put out by the TMP. As written the program will loop endlessly unless diverted by an external interrupt such as point A in *Figure 2.* Since it just missed the previous rising ALE edge it will not be until point B that the logic low INTR is read in. However, by then the CPU will have started execution of the first byte of the JMP 11 instruction. Since instructions are always finished once started, it will not be until point C that we begin interrupt service. At this point the next address would have been back at 11 but we now want to service the interrupt and push the stack. Stack pushing or popping takes 2 CPU cycles so the two address 11's shown following point C are dummies. Finally, we start interrupt service at point D by outputting address 003 and reading in the IN PORT instruction. Since the IN PORT instruction is only 1 byte long but takes 2 CPU cycles to execute, the address "4" at point E is a dummy and isn't really needed until point F when we read in the RETR instruction. Like IN PORT, RETR is a 1 byte instruction that takes 2 CPU cycles to execute. Therefore, the address "5" at point G is redundant. Upon returning from subroutine we immediately push the stack again (point H) since the interrupt is still there. Note that we immediately push the stack and do not execute the JMP at 11. Once more we go through the interrupt service routine but this time the interrupt ends at point I. Since it missed the preceding rising ALE edge where it was still seen as a logic low, we will immediately execute another interrupt service routine as shown. Finally, at point J as we prepare to return from service, INTR will be seen as a logic high and from point K onward execution will proceed normally.

When enabling and disabling interrupts, the rules for when you will and will not service them are predicated on the latest sampled interrupt level and last instruction executed. This is illustrated by the example program of *Figure 3* and instruction execution sequences of *Figure 4.* As shown in *Figure 4a,* the interrupt goes low at point A and will be sampled at the rising ALE of point B. However, since the current executing instruction (DIS XI at location 13) must be completed before starting interrupt service, the interrupt will be

```
ADDRESS     OPCODE      MNEMONIC
  000         04        JMP 010          ;RESET VECTOR
  001         10
  002
  003         E1        IN PORT          ;EXTERNAL INTERRUPT VECTOR
  004         93        RETR
  005
  006
  007
  008
  009
  00A
  00B
  00C
  00D
  00E
  00F
  010         05        EN XI            ;MAIN PROGRAM
  011         04        JMP 001
  012         11
  013
  014
  015
```

FIGURE 1. INTR Service Timing Example Program



TL/DD/6972–1

FIGURE 2. INTR Service Timing

locked out. Execution continues unperturbed until the interrupt is re-enabled with an EN XI from location 11, point F. Although the interrupt went logic high at point E it was still sampled as a logic low at point D.

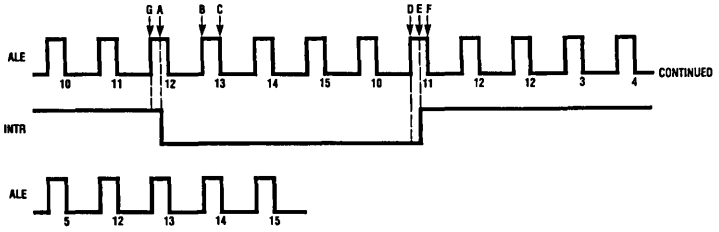
Therefore, after executing the EN XI at location 11, interrupt service will commence as shown. If the interrupt had gone logic high before point D it would have been sampled high and no interrupt service would have been performed.

Returning to the missed interrupt at point A, if the interrupt low had come in time to be sampled at point G, the instruction at 12 would have been the last one executed before interrupt service started as demonstrated in *Figure 4b*.

Although describing the external interrupt, all of the service sequences presented may be directly applied to TMP internal interrupts.

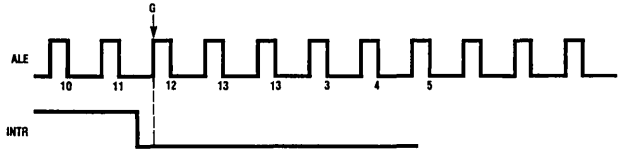
```
ADDRESS     OPCODE      MNEMONIC
  000         04        JMP 010          ;RESET VECTOR
  001         10
  002
  003         00        NOP              ;EXTERNAL INTERRUPT VECTOR
  004         93        RETR
  005
  006
  007
  008
  009
  00A
  00B
  00C
  00D
  00E
  00F
  010         00        NOP              ;MAIN PROGRAM
  011         05        EN XI
  012         00        NOP
  013         15        DIS XI
  014         04        JMP 010
  015         10
  016
```

FIGURE 3. INTR Enable/Disable Timing Example Program



TL/DD/6972-2

FIGURE 4a. INTR Enable/Disable Timing



TL/DD/6972-3

FIGURE 4b

TMP Row and Attribute Table Lookup Operation

National Semiconductor
Application Note 354
James Murashige

This note describes in detail the operation of the TMP Attribute Demo Program - TAD. Although a short program, it nicely demonstrates row table lookup operation in the TMP while at the same time putting out a visual display of the various video attributes available in the chip. While this display management approach is much more involved than normal sequential lookup mode, it is necessary when attemping to do fast screen updates or line editing with the TMP.

The hardware environment for which the program was written is the TMP Demo board. Appropriate references to and descriptions of the hardware will be made as necessary. For those who have not seen it, the net function of the program is to put up and manage a single frame of video data. In the top half of the display the same message is repeated 5 times but each time with a different set of attributes. In the lower half of the display are 4 rows representing the 128 possible block graphics patterns. All of the attribute effects displayed are achieved by updating the internal AL0 attribute latch at the end of each display row. At the same time a message table lookup is performed in order to obtain the appropriate character string that will work with the new attribute set selected.

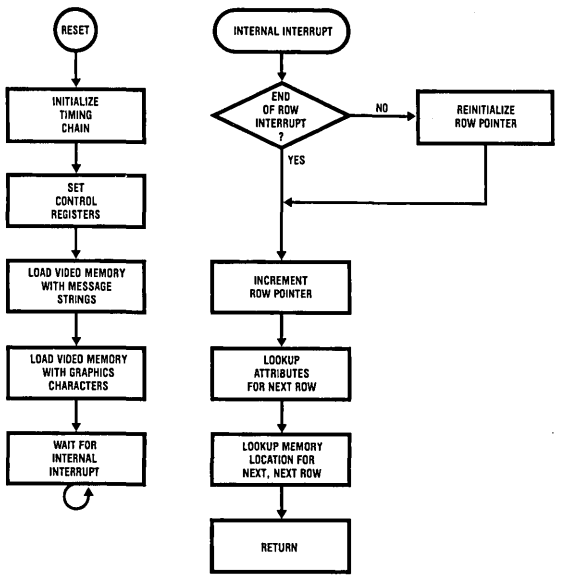
The flowchart for the program is shown in *Figure 1.* As you can see, the program essentially consists of initialization and waiting for and servicing video interrupts to manage the screen display. Initialization starts at BEGIN with the Vertical Interrupt Register and Timing Chain being loaded first. The Vertical interrupt is used for end of frame synchronization and is set to activate after the 27th row. The Timing Chain is loaded as follows:

TCP 0	Horizontal Length	=	104
1	Characters/Row	=	80
2	Horizontal Sync Begin	=	84
3	Horizontal Sync End	=	100
4	Character Height	=	10
	Extra Scans/Frame	=	2
5	Vertical Length	=	27
6	Vertical Blank	=	25
7	Vertical Sync Begin/End	=	7,3
8	Status Row Begin	=	31
9	Blink Rate/D.C.	=	F4H
10	Graphics Column Register	=	30H
11	Graphics Row Register	=	36H
12	Underline Size Register	=	89H
13	Cursor Size Register	=	09H

Given these values, one can ascertain that the display is 80 columns across and 25 rows tall. The character cell height is 10 scan lines and no status line will be displayed. The character underline is the bottom most scan line in a cell and the cursor occupies an entire cell. The partitioning of the block graphics cells is as follows:

```
0011100
0011100
0011100
2233344
2233344
5566655
5566655
5566655
5566655
```

TAD Flowchart



TL/C/5729–1

FIGURE 1

Following timing chain initialization various system registers are set to configure the chip to operate in its hardware environment. The video memory is a 2kX8 NMC2116 located between addresses 000-7FF. The crystal dot clock is 12 Mhz allowing us to use divide by 1 to generate the CPU clock. Accordingly the SCR is set to 24H (SB8-15 address output only, cell width = 7, divide by 1 for CPU clock, row table lookup operation). RAM Bank 0 is selected and HOME, BEGD, RA/RO are cleared. ENDD and CURS are set to 7FFFH and AL1 is set to FFH (no attributes selected). Video display memory (80×25 char) is then cleared out by storing spaces at all of the memory locations. Along with the spaces, attribute latch 1 is specified to be used. Video is then turned on by setting the VCR to 21H (normal alphanumeric display, internal attribute latch operation, normal video).

Next, the message tables are built up in the video memory. By updating the attribute latch AL0 each row, the entire screen display can be constructed from the 7 message rows stored in memory. Each of the message rows consist of 80 consecutive characters and are called up for display by loading the HOME register with the address of the first character in the row. The background characters in each of the rows are the spaces previously stored. Each of the display characters stored use attribute latch AL0 which is updated each row. The first row (0-79) consists entirely of spaces to provide us with a blank display row. The second row (80-159) has the message "tmp does it BETTER!" for normal and double high display. The third row (160-239) contains "ttmmpp ddooeess iitt BBEETTTTEERR!!" for double wide and double size display. Rows 4–7 contain 32 block graphics characters per row for a total of 128 patterns. The 128 characters stored are merely all binary combinations of the low 7 data bits in ascending order. The 32 characters in each row are stored in every other memory location to achieve a blank space between characters. For all of the message rows, data is positioned to give a centered display on the screen.

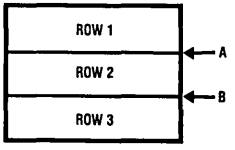
With initialization accomplished, we set the interrupt mask, re-enable interrupts and wait for a video interrupt.

Video display management is performed by the internal interrupt service routine located at 007 and consists of updating the HOME register and AL0 at the end of each display row. To accomplish this, a row counter (R3) is used as a pointer into the data lookup tables which follow the interrupt service routine. The R3 row counter is incremented on each End of Row interrupt or preset and incremented on a re-synching Vertical Interrupt.

Because the next row pointers are pipelined in the video memory controller, an understanding of End of Row and Vertical Interrupt operation is necessary in order to correctly set up the interrupt service routine and lookup tables. In table lookup mode, the Current Row Start Register (CRSR), which is a pointer to the first character address in a row, is automatically reloaded from the HOME register after the display of the last scan line in a row, a few characters into horizontal blanking. The timing of the CRSR reload when operating in sequential lookup mode is the same but in this case the pointer is advanced by the character width of the display row. It is the reloading of CRSR either in sequential or table lookup modes that generates the End of Row interrupt. The duration of the signal is ⅓ CPU cycle making it a one time event each row. The End of Row interrupt register bit is cleared when a reload of HOME, i.e., MOV HOME, A is executed. A simple example will illustrate the pipelining involved. In *Figure 2*, at the end of Row 1 (Point A) an EOR interrupt is generated. In preparation for this event HOME should have been loaded with the starting address of ROW 2 since the interrupt is generated when CRSR reloads from HOME. In service of the EOR, the program would load HOME with the starting address of ROW 3 in preparation for the EOR interrupt at Point B. However, notice that we have an entire row time from A to B to do the HOME reload. Finally note that EOR's are generated at the end of all rows except those blanked during vertical blanking. Vertical Interrupt operates with the same timing as End of Row except that it is specified to occur at the end of a particular row designated by the Vertical Interrupt Register. The row that it is specified to occur on must be < = Vertical Length Register (timing chain rows are counted starting from 0). Otherwise, it will never occur since the row counter will never count up that far. Usually Vertical Interrupt is specified to occur on a row blanked during vertical blanking so that it may be used as a frame sync signal.

Returning to TAD, *Figure 3* shows the interrupt positioning for all of the rows on the screen including the blanked ones. There are 25 displayed rows and 2 blanked ones in a frame for a total of 27. In addition, there are 2 extra scan lines which may be ignored as far as interrupt operation is concerned. Vertical Interrupt is set to occur at the end of the last row in the frame as shown. Row pointer operation for rows 2 to 24 is pipelined as described in *Figure 2.* At the end of ROW 24 (point E) the CRSR will be loading the pointer to ROW 25 and the interrupt service will load HOME with the pointer to ROW 1. At the end of ROW 25 (point F) the CRSR will load the pointer to ROW 1 and save it for the next frame. Since no EOR's are generated during vertical blanking, CRSR will remain static until ROW 1. At this point, it doesn't matter what the interrupt service loads into HOME and AL0 since the Vertical Interrupt at ROW 27 will reset the row counter and perform a new lookup for HOME and AL0. A Vertical Interrupt will not do a CRSR load, thus the pointer to ROW 1 will be preserved. At Vertical Interrupt, the row counter will be reset to 0 and we will want to do a pointer lookup for ROW 2 in preparation for the CRSR load at the end of ROW 1 (point A). Correspondingly, the row pointer lookup tables are organized 2 to 25, 1. Since the attribute latches aren't pipelined, the AL0 lookup table is arranged 1 to 25 since the new attribute set will be needed immediately for the display of the next row.

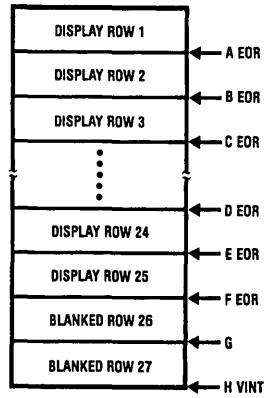
Row Table Lookup Pipelining



TL/C/5729–2

FIGURE 2

TAD Interrupt Positioning



TL/C/5729-3

FIGURE 3

TMP Attribute Demo Program

```
 1
 2
 3                .TITLE MAIN, 'TMP ATTRIBUTE DEMO - TAD'
 4
 5                ;                              James Murashige 10/05/83
 6                ;This program displays the various character attributes available with
 7                ;the TMP by dynamically updating the attribute latch each display row.
 8                ;In addition it uses End of Row and Vertical interrupts to perform row
 9                ;table lookup screen refreshing.
10
11       0000     LINE 1   =     0      ;LINE 1 START, ALL BLANKS
12       0050     LINE 2   =    80      ;LINE 2 START, NORMAL MESSAGE
13       00A0     LINE 3   =   160      ;LINE 3 START, DOUBLE WIDE MESSAGE
14       00F0     LINE 4   =   240      ;LINE 4 START, FIRST GRAPHICS LINE
15       0140     LINE 5   =   320      ;LINE 5 START, SECOND GRAPHICS LINE
16       0190     LINE 6   =   400      ;LINE 6 START, THIRD GRAPHICS LINE
17       01E0     LINE 7   =   480      ;LINE 7 START, FOURTH GRAPHICS LINE
18
19
20
21       0000     . = 00                ;START AT PROGRAM LOCATION 0
22
23 0000 0468      RESET;  JMP BEGIN     ;VECTOR TO RESET CODE
24
25       0003     . = 03
26
27                EXI:                  ;VECTOR TO EXTERNAL INTERRUPT PROCESSING
28
29       0007     . = 07
30
31                INI:                  ;VECTOR TO INTERNAL INTERRUPT PROCESSING
32
33 0007 8C                MOV A, INTR     ;READ INTERRPUT REGISTER
34 0008 320C              JBI EOR         ;HAVE AN EOR INTERRPUT
35 000A BBFF              MOV R3, #0FF    ;VINT INTERRUPT
36 000C 1B       EOR:     INC R3          ;INCREMENT TO DO NEXT ROW
37 000D 234F              MOV A, #ATT0    ;GET ATTRIBUTE LATCH 0
38 000F 6B                ADD A, R3
39 0010 B3                MOVP A, @A
40 0011 3C                MOV ALO, A      ;LOAD ATTRIBTE LATCH 0
41
42 0012 231D              MOV A, #HOMHIG  ;GET HOME HIGH ORDER BYTE
43 0014 6B                ADD A, R3
```

TMP Attribute Demo Program (Continued)

```
 44 0015 B3              MOVP A, @A
 45 0016 C2              MOV HACC, A
 46 0017 2336            MOV A, #HOMLOW ;GET HOME LOW ORDER BYTE
 47 0019 6B              ADD A, R3
 48 001A B3              MOVP A, @A
 49 001B 8A              MOV HOME, A     ;LOAD HOME
 50 001C 93              RETR
 51              .FORM
 52
 53              ;HOME HIGH ORDER BYTE LOOKUP TABLE
 54
 55 001D 00      HOMHIG: .BYTE 0             ;ROW 2
 56 001E 00              .BYTE 0             ;ROW 3
 57 001F 00              .BYTE 0             ;ROW 4
 58 0020 00              .BYTE 0             ;ROW 5
 59 0021 00              .BYTE 0             ;ROW 6
 60 0022 00              .BYTE 0             ;ROW 7
 61 0023 00              .BYTE 0             ;ROW 8
 62 0024 00              .BYTE 0             ;ROW 9
 63 0025 00              .BYTE 0             ;ROW 10
 64 0026 00              .BYTE 0             ;ROW 11
 65 0027 00              .BYTE 0             ;ROW 12
 66 0028 00              .BYTE 0             ;ROW 13
 67 0029 00              .BYTE 0             ;ROW 14
 68 002A 00              .BYTE 0             ;ROW 15
 69 002B 00              .BYTE 0             ;ROW 16
 70 002C 00              .BYTE 0             ;ROW 17
 71 002D 00              .BYTE 0             ;ROW 18
 72 002E 01              .BYTE H(LINE5)      ;ROW 19
 73 002F 00              .BYTE 0             ;ROW 20
 74 0030 01              .BYTE H(LINE6)      ;ROW 21
 75 0031 00              .BYTE 0             ;ROW 22
 76 0032 01              .BYTE H(LINE7)      ;ROW 23
 77 0033 00              .BYTE 0             ;ROW 24
 78 0034 00              .BYTE 0             ;ROW 25
 79 0035 00              .BYTE 0             ;ROW 1
 80 0036 00              .FORM
 81
 82              ;HOME LOW ORDER BYTE LOOKUP TABLE
 83
 84 0036 00      HOMLOW: .BYTE 0             ;ROW 2 BLANK
 85 0037 50              .BYTE L(LINE2)      ;ROW 3 NORMAL
 86 0038 00              .BYTE 0             ;ROW 4 BLANK
 87 0039 A0              .BYTE L(LINE3)      ;ROW 5 DOUBLE WIDE
 88 003A 00              .BYTE 0             ;ROW 6 BLANK
 89 003B 50              .BYTE L(LINE2)      ;ROW 7 DOUBLE HIGH
 90 003C 50              .BYTE L(LINE2)      ;ROW 8 DOUBLE HIGH
 91 003D 00              .BYTE 0             ;ROW 9 BLANK
 92 003E A0              .BYTE L(LINE3)      ;ROW 10 DOUBLE SIZE
 93 003F A0              .BYTE L(LINE3)      ;ROW 11 DOUBLE SIZE
 94 0040 00              .BYTE 0             ;ROW 12 BLANK
 95 0041 A0              .BYTE L(LINE3)      ;ROW 13 DOUBLE SIZE
 96 0042 A0              .BYTE L(LINE3)      ;ROW 14 DOUBLE SIZE
 97 0043 00              .BYTE L(LINE1)      ;ROW 15 BLANK
 98 0044 00              .BYTE L(LINE1)      ;ROW 16 BLANK
 99 0045 F0              .BYTE L(LINE4)      ;ROW 17 GRAPHICS
100 0046 00              .BYTE L(LINE1)      ;ROW 18 BLANK
101 0047 40              .BYTE L(LINE5)      ;ROW 19 GRAPHICS
102 0048 00              .BYTE L(LINE1)      ;ROW 20 BLANK
103 0049 90              .BYTE L(LINE6)      ;ROW 21 GRAPHICS
104 004A 00              .BYTE L(LINE1)      ;ROW 22 BLANK
105 004B E0              .BYTE L(LINE7)      ;ROW 23 GRAPHICS
106 004C 00              .BYTE L(LINE1)      ;ROW 24 BLANK
107 004D 00              .BYTE L(LINE1)      ;ROW 25 BLANK
108 004E 00              .BYTE L(LINE1)      ;ROW 1 BLANK
```

TMP Attribute Demo Program (Continued)

```
109                 .FORM
110
111                 ;ATTRIBUTE LATCH 0 LOOKUP TABLE
112
113 004F FF      ATT0:   .BYTE 0FF           ;ROW 1
114 0050 FF             .BYTE 0FF           ;ROW 2
115 0051 FF             .BYTE 0FF           ;ROW 3
116 0052 FF             .BYTE 0FF           ;ROW 4
117 0053 EF             .BYTE 0EF           ;ROW 5
118 0054 FF             .BYTE 0FF           ;ROW 6
119 0055 F7             .BYTE 0F7           ;ROW 7
120 0056 B7             .BYTE 0B7           ;ROW 8
121 0057 FF             .BYTE 0FF           ;ROW 9
122 0058 E7             .BYTE 0E7           ;ROW 10
123 0059 A7             .BYTE 0A7           ;ROW 11
124 005A FF             .BYTE 0FF           ;ROW 12
125 005B E2             .BYTE 0E2           ;ROW 13
126 005C 82             .BYTE 082           ;ROW 14
127 005D FF             .BYTE 0FF           ;ROW 15
128 005E FF             .BYTE 0FF           ;ROW 16
129 005F 7F             .BYTE 07F           ;ROW 17
130 0060 FF             .BYTE 0FF           ;ROW 18
131 0061 7F             .BYTE 07F           ;ROW 19
132 0062 FF             .BYTE 0FF           ;ROW 20
133 0063 7F             .BYTE 07F           ;ROW 21
134 0064 FF             .BYTE 0FF           ;ROW 22
135 0065 7F             .BYTE 07F           ;ROW 23
136 0066 FF             .BYTE 0FF           ;ROW 24
137 0067 FF             .BYTE 0FF           ;ROW 25
138                 .FORM
139
140                 ;START OF INITIALIZING CODE
141
142 0068 15      BEGIN; DIS XI              ;INTERRUPTS OFF FOR NOW
143 0069 35             DIS II
144 006A 65             STOP T
145 006B 231A           MOV A, #26
146 006D A2             MOV VINT, A
147 006E 27             CLR A              ;SET UP TIMING CHAIN FOR DEMO BOARD
148 006F 87             MOV TCP, A
149 0070 2367           MOV A, #103        ;HORIZONTAL LENGTH
150 0072 B7             MOV @TCP, A
151 0073 234F           MOV A, #79         ;CHARACTERS/ROW
152 0075 B7             MOV @TCP, A
153 0076 2353           MOV A, #83         ;HORIZONTAL SYNC BEGIN
154 0078 B7             MOV @TCP, A
155 0079 2363           MOV A, #99         ;HORIZONTAL SYNC END
156 007B B7             MOV @TCP, A
157 007C 2391           MOV A, #091        ;CHARACTER HEIGHT/EXTRA SCANS
158 007E B7             MOV @TCP, A
159 007F 231A           MOV A, #26         ;VERTICAL LENGTH
160 0081 B7             MOV @TCP, A
161 0082 2318           MOV A, #24         ;VERTICAL BLANK
162 0084 B7             MOV @TCP, A
163 0085 2362           MOV A, #062        ;VERTICAL SYNC BEGIN/END
164 0087 B7             MOV @TCP, A
165 0088 231E           MOV A, #30         ;STATUS ROW BEGIN
166 008A B7             MOV @TCP, A
167 008B 23F4           MOV A, #0F4        ;BLINK RATE
168 008D B7             MOV @TCP, A
169 008E 2330           MOV A, #030        ;GRAPHICS COLUMN REGISTER
170 0090 B7             MOV @TCP, A
171 0091 2336           MOV A, #036        ;GRAPHICS ROW REGISTER
172 0093 B7             MOV @TCP, A
173 0094 2389           MOV A, #089        ;UNDERLINE SIZE REGISTER
```

TMP Attribute Demo Program (Continued)

```
174 0096 B7              MOV @TCP, A
175 0097 2309            MOV A, #009      ;CURSOR SIZE REGISTER
176 0099 B7              MOV @TCP, A
177
178
179              .FORM
180 009A 2324            MOV A, #024      ;SET SYSTEM CONTROL REGISTER
181 009C 55              MOV SCR, A       ;8 BI,7 DOTS, DIVIDE 1, TABLE LOOKUP
182
183 009D C3              SEL RB0          ;SELECT RAM BANK 0
184 009E 27              CLR A            ;SET RAM POINTERS
185 009F C2              MOV HACC, A
186 00A0 8A              MOV HOME, A
187 00A1 0D              MOV BEGD, A
188 00A2 88              MOVL R0, A       ;CLEAR MEMORY POINTER
189
190 00A3 237F            MOV A, #07F
191 00A5 C2              MOV HACC, A
192 00A6 23FF            MOV A, #0FF
193 00A8 0C              MOV ENDD, A
194 00A9 8B              MOV CURS, A
195 00AA 3D              MOV AL1, A       ;NO ATTRIBUTES FOR LATCH 1
196
197              ;CLEAR OUT MEMORY
198
199 00AB BD19             MOV R5, #25     ;DO 25 ROWS
200 00AD BA50             MOV R2, #80     ;DO 80 CHARACTERS PER ROW
201 00AF 23A0             MOV A, #0A0     ;INITIALIZE FOR A SPACE, ATTRIBUTE LATCH 1
202
203 00B1 80      LOOP:    MOVX @R0, A     ;STORE A CHARACTER
204 00B2 38               INCL R0         ;INCREMENT POINTER
205 00B3 EAB1             DJNZ R2, LOOP   ;TEST IF ROW DONE
206 00B5 BA50             MOV R2, #80
207 00B7 EDB1             DJNZ R5, LOOP   ;TEST IF SCREEN DONE
208
209 00B9 2321             MOV A, #021     ;SET VCR FOR INTERNAL ATTRIBUTES
210 00BB 45               MOV VCR, A      :INTERNAL CHARACTER GENERATOR
211
212
213              ;FIRST LINE ARE ALL BLANKS, SECOND LINE HAS SINGLE SPACING MESSAGE
214 00BC 2300             MOV A, #H(LINE2+30)  ;SET R0 POINTER TO FIRST LINE
215 00BE C2               MOV HACC, A
216 00BF 236E             MOV A, #L(LINE2+30)
217 00C1 88               MOVL R0, A
218 00C2 BAE0             MOV R2, #L(MSG1)  ;SET R2 TO MESSAGE #1
219 00C4 BB13             MOV R3, #19      ;SET R3 TO MESSAGE LENGTH
220 00C6 FA      DISP1:   MOV A, R2
221 00C7 B3               MOV A, @A        ;DISPLAY NORMAL MESSAGE
222 00C8 80               MOVX @R0,A
223 00C9 38               INCL R0
224 00CA 1A               INC R2
225 00CB EBC6             DJNZ R3, DISP1
226              .FORM
227              ;THIRD LINE HAS DOUBLE WIDE MESSAGE
228 00CD 98               MOVL A, R0                ;SET R0 POINTER
229 00CE 0334             ADD A, #(31 + 21)        ;LINES3 + 21
230 00D0 88               MOVL R0, A
231 00D1 BAE0             MOV R2, #L(MSG1)
232 00D3 BB13             MOV R3, #19
233 00D5 FA      DISP2:   MOV A, R2
234 00D6 B3               MOVP A, @A       ;DISPLAY DOUBLE WIDE
235 00D7 80               MOVX @R0, A
236 00D8 38               INCL R0
237 00D9 80               MOVX @R0, A
238 00DA 38               INCL R0
```

TMP Attribute Demo Program (Continued)

```
239 00DB 1A             INC R2
240 00DC EBD5           DJNZ R3, DISP2
241 00DE 04F3           JMP FOURTH
242
243 00E0 74     MSG1:   .BYTE 'tmp does it BETTER!'
244
245             ;FOURTH LINE STARTS GRAPHICS CHARACTERS DISPLAY
246 00F3 98     FOURTH: MOVL A, R0
247 00F4 031D           ADD A, #(21 + 8)     ;LINE4 + 8
248 00F6 88             MOVL R0, A
249 00F7 BB04           MOV R3, #4          ;DO 4 LINES
250 00F9 BA20           MOV R2, #32         ;DO 32 GRAPHICS CHARACTERS PER LINE
251 00FB 2300           MOV A, #000          ;ATTRIBUTE LATCH 0 SELECTED
252 00FD 2400           JMP BLOOP
253
254      0100     . = 0100
255
256 0100 80     BLOOP:  MOVX @R0, A         ;STORE CHARACTER
257 0101 38             INCL R0
258 0102 38             INCL R0
259 0103 17             INC A
260 0104 EA00           DJNZ R2, BLOOP
261
262 0106 BA20           MOV R2, #32         ;INITIALIZE FOR NEW ROW
263 0108 AC             MOV R4, A           ;TEMPORARY SAVE A
264 0109 98             MOVL A, R0
265 010A 0310           ADD A, #(8+8)       ;POINT TO NEXT LINE
266 010C 88             MOVL R0, A
267 010D FC             MOV A, R4           ;RESTORE A
268 010E EB00           DJNZ R3, BLOOP      ;CONTINUE IF NOT THROUGH
269
270             ;REENABLE INTERNAL INTERRUPTS AND MASK OFF UNUSED ONES
271 0110 2303           MOV A, #03
272 0112 82             MOV MASK, A
273 0113 25             EN II               ;REENABLE INTERNALS
274 0114 2414   PAU:    JMP PAU             ;WAIT FOR A VIDEO INTERRUPT
275             END
ATTO    004F    BEGIN   0068    BLOOP   0100    DISP1   00C6
DISP2   00D5    EOR     000C    EXI     0003 *  FOURTH  00F3
HOMHIG  001D    HOMLOW  0036    INI     0007 *  LINE1   0000
LINE2   0050    LINE3   00A0    LINE4   00F0    LINE5   0140
LINE6   0190    LINE7   01E0    LOOP    00B1    MSG1    00E0
PAU     0114    RESET   0000 *

NO ERROR LINES
  272 ROM BYTES USED

SOURCE CHECKSUM=CF60
OBJECT CHECKSUM=0576

INPUT FILE A: TAD. MAC
LISTING FILE A: TAD. PRN
OBJECT FILE A: TAD. LM
```

TMP - Dynamic RAM Interfacing

National Semiconductor
Application Note 355
James Murashige

TMPs Interface easily and directly to dynamic RAMs as illustrated in the basic TMP system schematic of *Figure 1*. In addition to providing the necessary Read/Write cycle control, the TMP will also automatically refresh the memories through the video controller, further easing interface requirements.

The circuitry to the right of the TMP provides program memory interfacing and I/O support while to the left lie the dynamic video RAM circuits. The memory width shown here is 8 bits although 16 bits can easily be accommodated. Using the 64K × 1 dynamic RAMs shown the entire video memory space is filled with RAM. However, by using a slightly modified addressing configuration smaller memroy chips could be substituted.

The requisite dynamic RAM control signals $\overline{RAS}$, $\overline{CAS}$ and $\overline{WE}$ are generated directly from the system bus control signals RAM ALE, $\overline{RAM\ RD}$ and $\overline{RAM\ WR}$. RAM ALE is used directly as $\overline{RAS}$ while $\overline{RAM\ WR}$ serves as $\overline{WE}$. $\overline{CAS}$ is the logical AND of $\overline{RAM\ RD}$ and $\overline{RAM\ WR}$. The 16 system bus bits are multiplexed down to the 8 bit RAM address vector by the two 74LS157's under the control of the RAM ALE. As configured, the row and column addresses strobed in are SB0-7 and SB8-15 respectively.

With the configuration shown, the pertinent TMP Read and Write cycle timing parameters for *Figure 2* are listed in Table 1. Going through the table one sees that the TMP easily interfaces to 150 ns access RAMs and will routinely work with 200 ns RAMs. The four parameters which may be a tight squeeze for 200 ns RAMs are:

1. t_{RAC}— Access Time from $\overline{RAS}$ is max 150 ns, typ 220 ns. This is a basic access time requirement which necessitates fast parts.
2. t_{RAH}— Row Address Hold Time is min 10 ns, typ 15 ns. This parameter is entirely dependent on the switching speed of the 74LS157.
3. t_{RCD}— $\overline{RAS}$ to $\overline{CAS}$ Delay Time is min 10 ns, typ 50 ns. This parameter isn't too critical since most dynamic RAMs internally gate the $\overline{CAS}$ signal should it come along too early.
4. t_{RP}— $\overline{RAS}$ Precharge Time is min 100 ns, typ 135 ns. Since $\overline{RAS}$ is actually the RAM ALE signal t_{RP} is the high time of RAM ALE.

However, rather than getting faster RAMs one could also meet spec by running the TMP CPU slower, thereby stretching out the allowable access time.

Since the TMP video controller will regularly and automatically access video memory in order to obtain characters for display, one may have dynamic RAM refreshing performed automatically by making sure that the required number of consecutive address locations (ROW Addresses) are accessed in the alloted time. Typically this is 128 ROW addresses in 2 ms.

For example, in a typical system we may have an 80 column by 25 row display with each row consisting of 10 scan lines. Each scan line has a period of 60.67 us. The vertical blank period consists of 25 scan lines for a total duration of 1.52 ms. Assuming sequential rather than table lookup operation, 80 consecutive character addresses are accessed each scan line and a 160 consecutive character addresses are accessed every 2 rows; more than enough to refresh all of the $\overline{RAS}$ rows. Of course one must be sure that the memory addresses of any two consecutive rows encompass all 128 possible $\overline{RAS}$ addresses. In the middle of the screen the worst case refresh period is 11 scan lines (.667 ms), since to do 160 consecutive addresses requires one complete row plus the first scan line of the next row. At the bottom of the screen the refresh period must also include the vertical blank time since no video characters are accessed then. In this case refresh stretches out to a worst case 2.184 ms.

Although in this example we exceeded the 2 ms refresh period, there are a number of things that we could do to get things back into spec. For example, we could cut down on vertical blank time, use memory chips with longer refresh periods, or have the CPU refresh video memory during vertical retrace. Taking the case of using different memory chips, another popular refreshing arrangement is 256 row addresses in 4 ms. In the middle of the screen this gives us a worst case period of 10 + 10 + 10 + 1 scan lines or 31 × 60.67 us = 1.88 ms. Adding in the vertical blanking period the absolute worst case refresh delay is 1.88 + 1.52 = 3.4 ms. Of course in this arrangement, making sure that any four consecutive rows encompass all 256 RAS addresses is much more difficult.

When operating in pixel mode meeting refresh requirements isn't as difficult since each scan line will access a different set of consecutive RAM addresses.

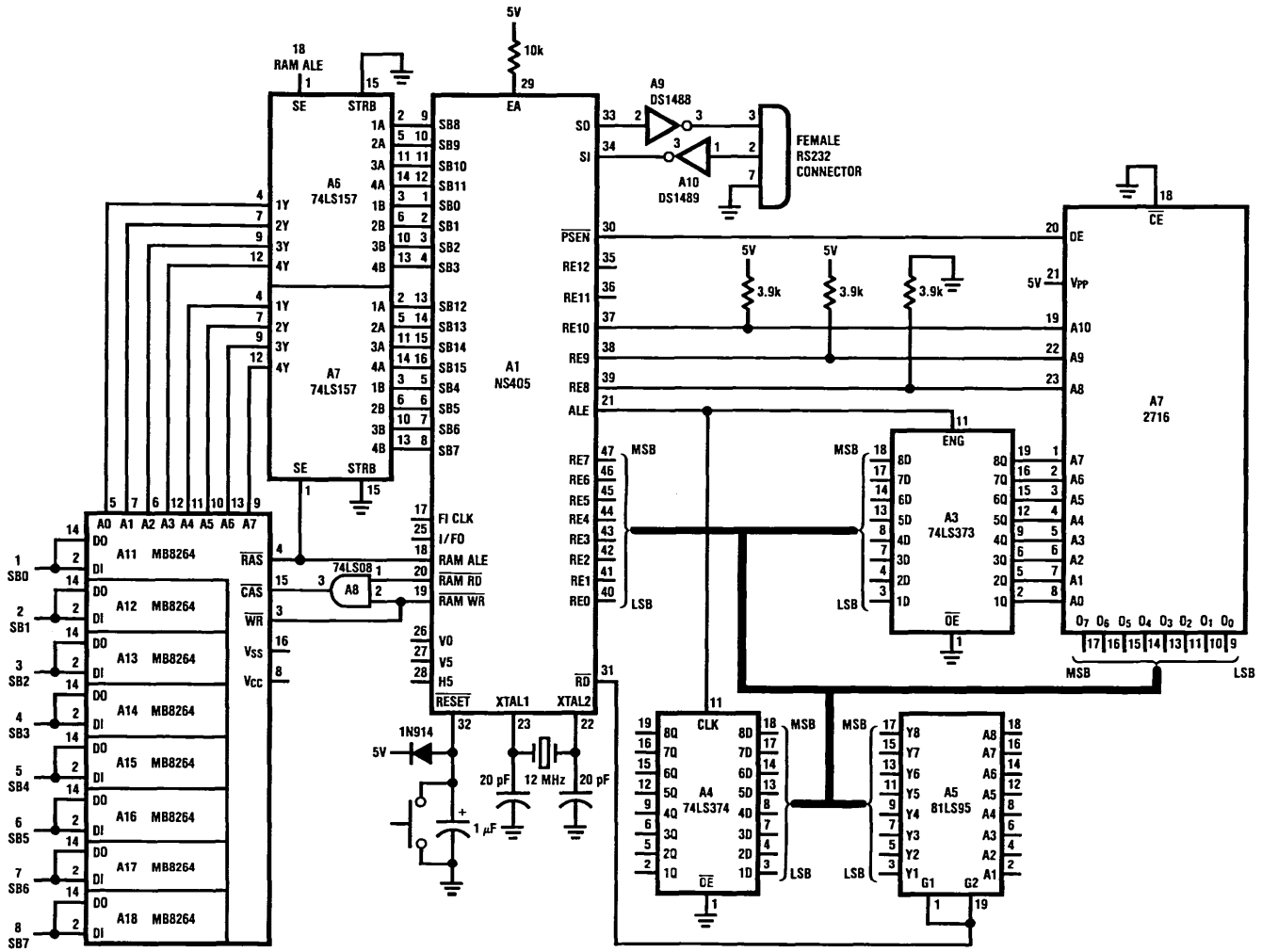
Returning to the circuit of *Figure 1*, we have assumed that SB0-7 are multiplexed address/data while SB8-15 output addresses only. Since the RAM addresses are latched in 8 bits at a time there is no need for a separate latch for SB0-7 since all 8 bits are clocked in on the falling ALE edge. However, when operating with smaller 8K or 16K RAMs where only 7 bits are clocked in at a time, latching arrangements for SB7 must be made. An example of this is shown in *Figure 3* where bits SB0-7 are all latched by the 74LS373.

Normally I/O registers, as well as other memory banks, will also be memory mapped into the 64K video RAM space. In order to do this some sort of chip enabling scheme must be worked out since the dynamic RAMs have no direct enable control. One possibility is shown in *Figure 4* where the RAM bank $\overline{CAS}$ and $\overline{WE}$ are disabled unless selected by the 74LS138 decoder. In this way the RAM output drivers will remain TRI-STATE® and no data will be written unless the bank is selected. However, memory refreshing as controlled by $\overline{RAS}$ will still be performed on each RAM bank.

By expanding on these basic examples a memory configuration for the TMP utilizing dynamic memories may be quickly and easily worked out.

TABLE 1. TMP Dynamic RAM Interface Timing 12 MHz CPU

Symbol	Parameter	Min	Typ	Max	Units
t_{AR}	Column Address Hold Time Referenced to $\overline{RAS}$	250	280		ns
t_{ASC}	Column Address Set Up Time–Dependent on Switching of 74LS157	25	35		ns
t_{ASR}	Row Address Set Up Time	20	90		ns
t_{CAC}	Access Time from $\overline{CAS}$		180	140	ns
t_{CAH}	Column Address Hold Time	140	250		ns
t_{CAS}	$\overline{CAS}$ Pulse Width	140	160		ns
t_{CP}	$\overline{CAS}$ Precharge Time	140	166		ns
t_{CRP}	$\overline{CAS}$ to $\overline{RAS}$ Precharge Time	100	136		ns
t_{CSH}	$\overline{CAS}$ Hold Time	250	280		ns
t_{CWL}	Write Command to $\overline{CAS}$ Lead Time	140	160		ns
t_{DH}	Data In Hold Time	160	175		ns
t_{DHR}	Data In Hold Time Referenced to $\overline{RAS}$	180	310		ns
t_{DS}	Data In Set Up Time	10	50		ns
t_{OFF}	Output Buffer Turn Off Delay	0		60	ns
t_{RAC}	Access Time from $\overline{RAS}$		220	150	ns
t_{RAH}	Row Address Hold Time–Dependent on Switching of 74LS157	10	15		ns
t_{RAS}	$\overline{RAS}$ Pulse Width	250	280		ns
t_{RC}	Random Read/Write Cycle Time	416			ns
t_{RCD}	$\overline{RAS}$ to $\overline{CAS}$ Delay Time	10	50		ns
t_{RCH}	Read Command Hold Time	100	175		ns
t_{RCS}	Read Command Set Up Time	100	175		ns
t_{RP}	$\overline{RAS}$ Precharge Time	100	135		ns
t_{RRH}	Read Command Hold Time Referenced to $\overline{RAS}$	100	175		ns
t_{RSH}	$\overline{RAS}$ Hold Time	140	160		ns
t_{RWL}	Write Command to $\overline{RAS}$ Lead Time	140	150		ns
t_{WCH}	Write Command Hold Time	140	150		ns
t_{WCR}	Write Command Hold Time Referenced to $\overline{RAS}$	160	275		ns
t_{WCS}	Write Command Set Up Time–Dependent on Delay of 74LS08	5	11		ns
t_{WP}	Write Command Pulse Width	140	150		ns

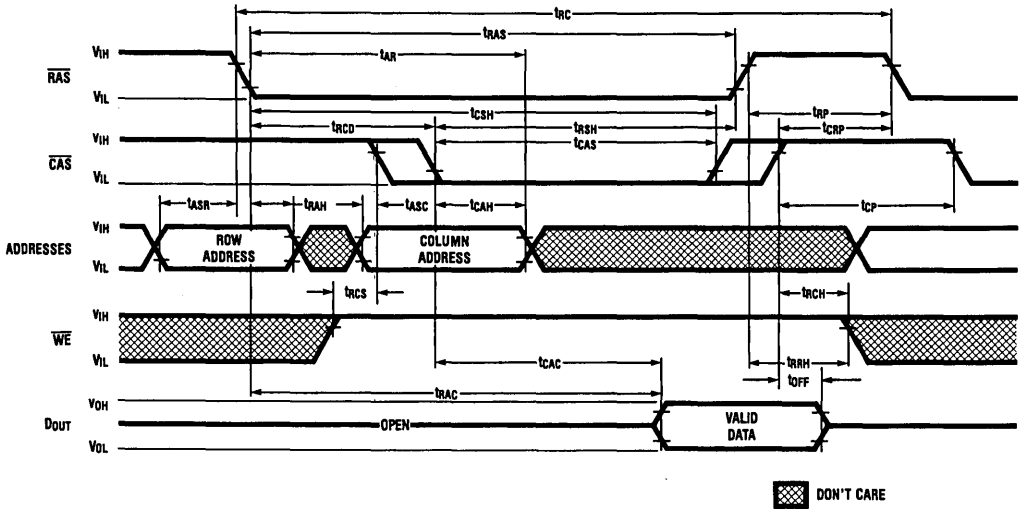


TL/C/5732–1

FIGURE 1. TMP with 64K Dynamic Memory

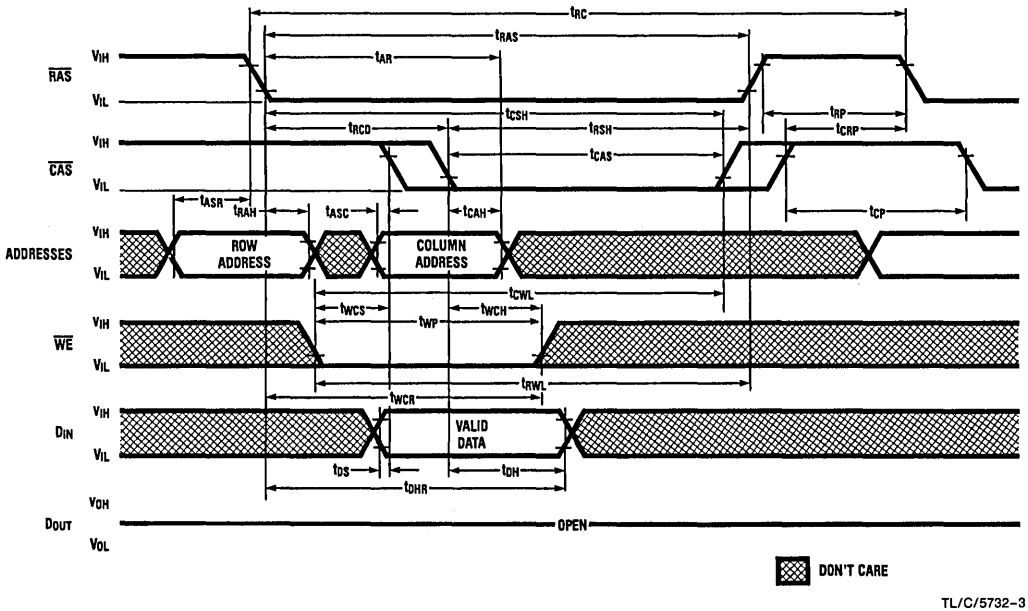
Timing Diagrams

Read Cycle Timing Diagram



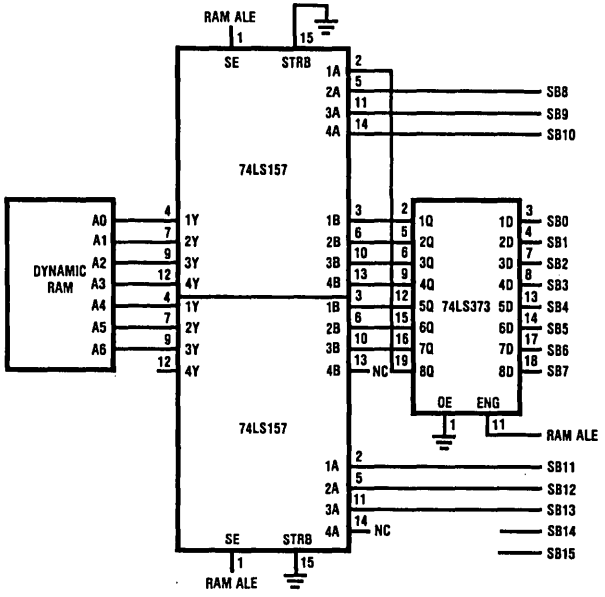
TL/C/5732-2

Write Cycle (Early Write)



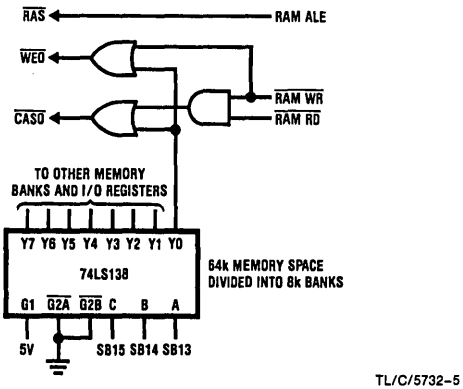
TL/C/5732-3

FIGURE 2.



TL/C/5732–4

FIGURE 3. TMP Address Multiplexing for 16K Dynamic RAMs



TL/C/5732–5

FIGURE 4. Chip Enabling Dynamic RAMs

TMP External Character Generation

National Semiconductor
Application Note 367
James Murashige

Built into the TMP video circuitry is the ability to access an external character generator to display custom FONT sets. In addition to the flexibility afforded by user selectable FONTs, by going "external" the number of different character patterns directly addressable is virtually limitless. On the other hand the disadvantages of going external are the additional hardware necessary to control data routing and the general need to use faster memories.

Figure 1 shows a minimum configuration with which to do external character generation. In the TMP, external character generation is selected through Video Control Register bits 6, 7 and is a cross between normal alphanumeric and pixel graphics display modes. Like normal alphanumeric mode the TMP sequences through the video memory address space based upon the screen format specification. But instead of routing the data through the internal character generator, it is treated as pixel data and directly inserted into the video dot stream. In effect what we are doing externally is duplicating the internal character generator ROM. In external mode video attributes are fully operational except for double height and block graphics.

Operation of the circuit shown is straight forward and follows a pipe-line approach. On a video data read the display memory address is output onto the system bus with the 8 low order bits being latched by the 74LS373. On the $\overline{\text{RAM}}$ $\overline{\text{RD}}$ signal the 2116 display RAM ouputs a data character onto the pipeline bus which is used to address the MM52116 character generator which in turn deposits the required pixel data onto the system bus so that it may be read in. The 2116 determines which character is to be looked up in the 52116 while the 74LS163 tells the character generator which row in the character we wish to look at. The 74LS163 is a counter which is appropriately clocked by the horizontal sync pulse so that we will advance each scan line to point to the next row in the character FONT. At the end of each screen row the counter must be cleared in preparation for the display of a new row. This is the function of the Scan Count Clear signal which is available as a multiplexed output on the RE11 pin. It is a low going signal which pulses for 1 scan line time during the last scan line in a screen ROW. Its timing is shown in Figure 2. Note that since the 74LS163 is a fully synchronous counter the clear input will not be accepted until the very last H-Sync clock pulse in the screen row. Because of the necessity to not clear the counter before all pixel data is brought in, nor to delay clocking lest the Scan Count Clear pulse be missed, the starting H-Sync clock edge must be postioned close to the start of horizontal blanking.

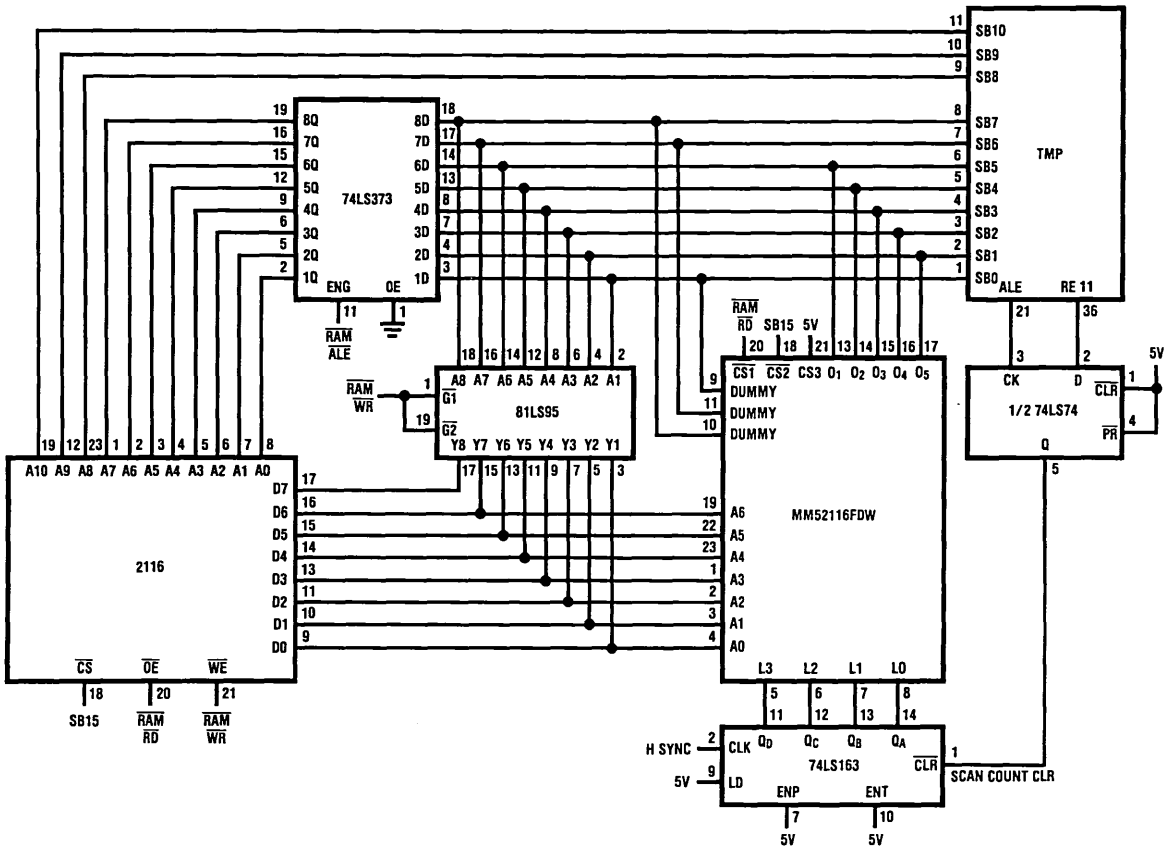
Continuing with the read operation, we see that video RAM is only accessed if SB15 is low, i.e., the lower 32K. Note that the 52116 used here contains 128 characters in a 5 × 7 FONT. Consequently, it has 5 data output lines connected to the system bus. The other three "dummy" lines shown connected are actually output bits which are always 0 by default, thus giving us blank spaces. There are two reasons why the character bits start on SB1. The first is that since everything brought in is considered pixel data, spaces between characters must be externally inserted. The second is that the video controller always brings in 8 bits even though the cell width can be defined to be 9 or 10. In these instances the 9th and 10th bits repeat what was encoded into the SB0 bit. As a result external characters can practically be at most 7 dots wide although the cells can be up to 8, 9, or 10 dots wide. Cell and/or character heights can be up to 16 lines tall as specified by the Character Scan Height Register.

On a video memory write, data is routed through the 81LS95, onto the pipe-line bus and into the 2116. Writing into the 2116 is controlled by $\overline{\text{RAM WR}}$ as shown. Ordinarily the MSB data bit is used for internal attribute latch selection and could be directly connected to the SB7 line if character cells were specified to at most be 7 dots wide. Otherwise SB7 will be needed for pixel generation as shown in Figure 1, thereby rendering internal attribute latch selection useless. In this case both internal attribute latches would have to be loaded with the same values. As shown here, 7 video RAM data bits are used to address the 128 possible characters in the 52116. If a larger character generator were available, additional data bits could be used to select from a larger character set. Since the TMP features a 16-bit multiplexed address/data bus, by using all 16 available data bits we could address 65,536 different character patterns

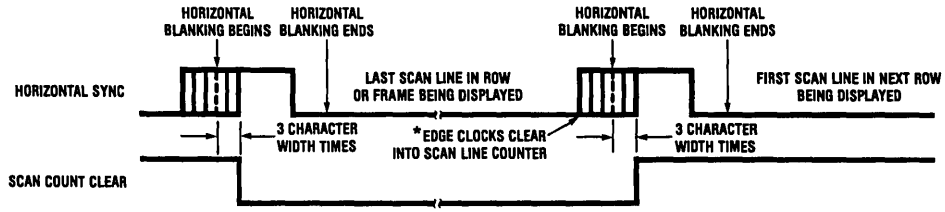
With the video data pipe-lined as shown, very fast memory circuits are required for external character generation. With a 12 MHz CPU clock, character pixel data must be available within a max of 220ns (typ. 300ns) after an address goes out. To accomplish this the character generator will typically have to be bipolar and the video RAM fast MOS. However, if faster memories are a problem, access times may be stretched out by slowing down the CPU clock since video RAM cycling is based on the CPU clock. For instance with a CPU clock of 8 MHz, access time stretches out to 385 ns max, 500 ns typ. If using the divide by 1.5 factor on the crystal to obtain the slower CPU clock, remember that due to system constraints the character cell MUST BE AT LEAST 8 DOTS WIDE. In Figure 1 the 2116 output enable is shown being driven by RAM RD. Although this may seem redundant and will slow things down (why not just leave the output enabled?) it is necessary in order to avoid bus conflict when doing a memory write operation.

By expanding on this basic circuit, numerous options such as external attributes, expanded character sets and dynamic RAM may be added to achieve the desired end system.

TMP EXTERNAL CHARACTER GENERATOR



SCAN COUNT CLEAR TIMING



TL/C/5731–2

* Edge must come before Scan Count Clear goes away but not before the video controller has brought in all necessary display information for the last scan line. Edge should not be more than 3 character widths from the beginning of blanking.

NS405 TMP Logic Analyzer

National Semiconductor
Application Note 369
James Murashige

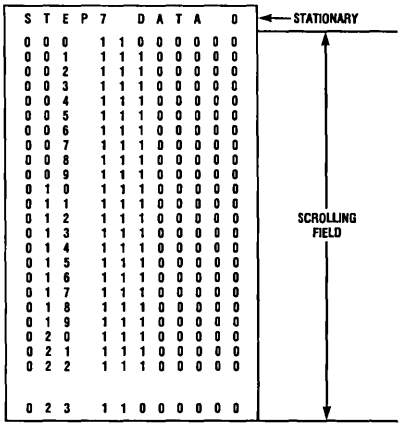
INTRODUCTION

The NS405 TMP is ideally suited for use in Test and Instrumentation equipment as the system or display controller. To demonstrate this, the following note describes how to turn the NS405 Demo Board into a simple 8 bit Logic State Analyzer. Featured in this system is a data capacity of 156 eight bit words, 21 μs data acquisition time, keyboard command entry, UP/DOWN rolling scroll and 24 line data display.

SYSTEM ARCHITECTURE

All of the necessary resources to build our system are available in a TMP Demo Board system when normally set up as a data terminal. Commands are entered through the attached ASCII encoded keyboard with data being strobed on the external interrupt. Data words are input through the switch configuration register SW2 by strobing the Light Pen interrupt. Video is output to the attached display monitor. The only real difference between our Logic Analyzer and the Data Terminal is the ROM software in U9 running the TMP. An overview of the system is shown in *Figure 1*.

In order to maximize the available 2k of video RAM, a display line length of 13 was chosen. This yields 157 lines of display information (157 × 13 = 2041), one of which is used to display title information. Thus our display data field consists of 156 lines of information, any 24 of which may be displayed at any given time. On each line is displayed the STEP number, followed by 2 spaces and 8 bits of 1 or 0 information. A typical display pattern is illustrated in *Figure 2*. By manipulating the pointer registers in the TMP DMA controller, the Title line is made to be stationary while the rest of the screen scrolls. This is accomplished by reversing the roles of the HOME register and Status Section SROW pointer. Specifically HOME points to the last row in memory which holds the title information while the status section is set to start after the display of the first row. Scrolling is accomplished by bumping the SROW pointer up or down 1 line width and checking for end of memory conditions.



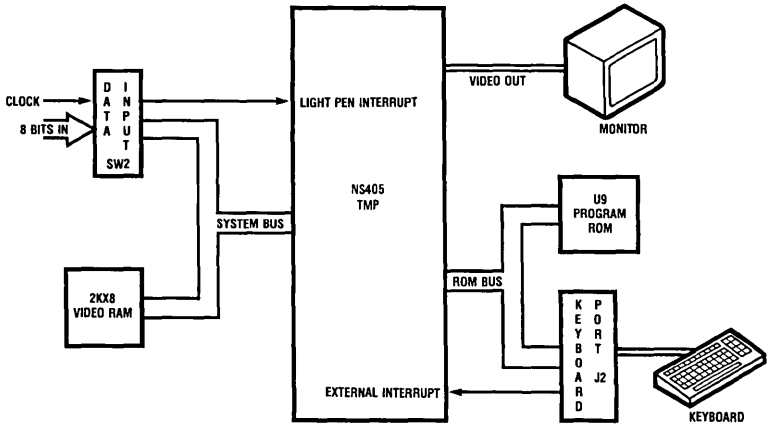
TL/DD/6970–2

FIGURE 2. TMP Logic Analyzer Screen Format

SYSTEM SOFTWARE

Since the system must rely on external events at several points before proceeding with processing, an interrupt driven approach was taken in structuring the software. A flowchart for the main program is shown in *Figure 3*. After system initialization there are 2 levels of processing associated with our logic analyzer operation. The first is a wait for an external interrupt signifying a new keyboard command. Referring to the keyboard service routine in *Figure 4*, the key is first read in and decoded as to function. In our simple system there are only 3 commands:

S or s = Start data acquisition
U or u = Scroll display up
I or i = Scroll display down



TL/DD/6970–1

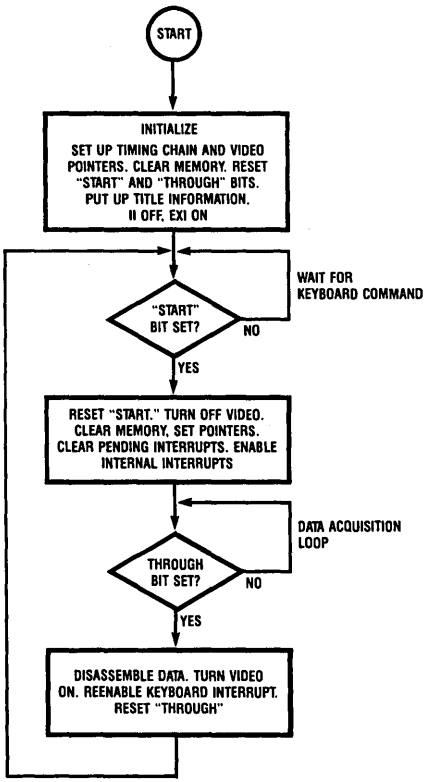
FIGURE 1. TMP Logic Analyzer (System Overview)

The scrolling functions are easily handled in the service routine by bumping the memory pointers and checking for an end of memory condition. A command to start data acquisition moves us to our second level of processing—the actual acquisition and display of data.

In both the keyboard and data acquisition interrupt service routines, flags F0 and F1 are used to pass system status back and forth from the main program. In this way the main program holds at major points while the service routines accomplish their functions. The data acquisition routine does nothing more than read data in from the SW2 port, store it in video memory and check a loop counter to see whether we have read in enough data. Since the Light Pen interrupt is being used, only high to low transitions will initiate an SW2 read. While very little is being done in data acquisition, it is time consuming because it's done in software. A count of instructions yields a worst case processing time of 21 μs between data strobes. In addition, since the data isn't latched it must remain stable until the actual read occurs. Following data acquisition, the stored data words are disassembled into their ASCII "1's" and "0's" patterns and the data entires numbered. With data acquistion completed, the program returns to await another keyboard command.

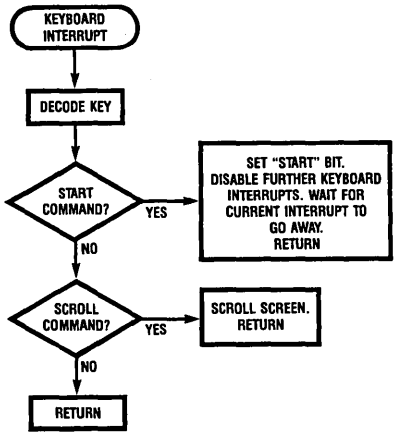
SUMMARY

As demonstrated, the NS405 is very effective as a display controller in a video instrumentation system. Certain functions, however, such as data acquisition are better left to dedicated hardware controllers. Nevertheless, the system as presented is still a very useful diagnostic tool. Through small enhancements to the hardware and software, features such as word recognition, number base conversion, wider data words and loop delay may readily be added.



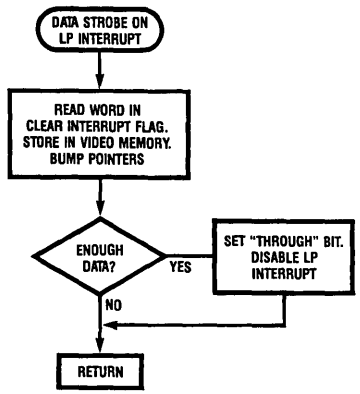
TL/DD/6970-3

FIGURE 3. TMP Demoboard Logic Analyzer Main Program



TL/DD/6970-4

FIGURE 4. TMP Demoboard Logic Analyzer Command Input Routine



TL/DD/6970-5

FIGURE 5. TMP Demoboard Logic Analyzer Data Acquisition Routine

```
1
2
3                .TITLE  MAIN,'TMP LOGIC ANALYZER DEMO'
4
5                ;                       James Murashige 2/09/84
6                ;This program turns the TMP Demo board into a simple 8 bit logic analyzer.
7                ;Command inputs are entered from the attached ASCII keyboard while data
8                ;acquisition takes place through the switch configuration socket, SW2.
9                ;The DIP switch may have to be unsoldered from the board.  Data is strobed
10               ;in with an external clock applied to Light Pen Interrupt on W8A.  Each time
11               ;data acqusition is started 156 words of 8 bits each are acquired and displayed.
12               ;Display is in the form of STEP location and the associated 8 bit 1's and 0's
13               ;pattern.
14               ;Commands are  S = Start data acqusition
15               ;              U = Scroll display up
16               ;              I = Scroll display down
17
18
19
20       07DF    LSTLIN = 07DF       ;START OF LAST LINE
21       07EB    MEMEND = 07EB       ;END OF MEMORY
22       07EC    STLIN  = 07EC       ;START OF TITLE LINE
23       0020    VON    = 020         ;VIDEO ON
24       0000    VOFF   = 000         ;VIDEO OFF
25
26
27
28       0000    . = 00              ;START AT PROGRAM LOCATION 0
29
30 0000 0452     RESET:  JMP BEGIN       ;VECTOR TO RESET CODE
31
32       0003    . = 03
33
34 0003 0412     EXI:    JMP KEY           ;VECTOR TO KEYBOARD COMMAND DECODE
35
36       0007    . = 07
37
38               INI:                 ;DATA STROBE INTERRUPT SERVICE
39 0007 8C               MOV A,INTR      ;CLEAR OUT INTERRUPT
40 0008 91               MOVX A,@R1      ;GET DATA CHARACTER
41 0009 80               MOVX @RO,A      ;STORE CHARACTER AWAY
42 000A 98               MOVL A,RO       ;BUMP RO POINTER
43 000B 6B               ADD A,R3
44 000C 88               MOVL RO,A
45 000D EA11             DJNZ R2,NOTRU   ;CHECK IF THROUGH
46
47 000F B5               CPL F1    ;YES THROUGH, SET INDICATOR BIT
48 0010 35               DIS II    ;DISABLE LP INTERRUPT
49
50 0011 93       NOTRU:  RETR        ;RETURN
51               .FORM
52               ;KEYBOARD COMMAND DECODE
53 0012 E1       KEY:     IN PORT          ;KEYBOARD DATA READ
54 0013 53DF             ANL A,#ODF      ;CONVERT LOWER TO UPPER CASE
55 0015 AA               MOV R2,A        ;SAVE COPY IN R2
56 0016 D353             XRL A,#'S'
57 0018 C627             JZ START        ;GOTO START
58 001A FA               MOV A,R2
59 001B D355             XRL A,#'U'
60 001D C62B             JZ UP       ;GOTO SCROLL UP
61 001F FA               MOV A,R2
62 0020 D349             XRL A,#'I'
63 0022 C63C             JZ DOWN  ;GOTO SCROLL DOWN
64 0024 A624     CKOFF:   JNXI CKOFF      ;NOT A VALID KEY & WAIT FOR EXI TO GO AWAY
65 0026 93               RETR         ;RETURN
66
67 0027 95       START:   CPL FO    ;START BIT SET
68 0028 15               DIS XI     ;DISABLE FURTHER KEYBOARD INTERRUPTS
69 0029 0424             JMP CKOFF
70
71 002B 99       UP:      MOVL A,R1        ;SCROLL UP
```

TL/DD/6970-6

```
 72 002C 030D           ADD A,#13       ;ADVANCE TO NEXT ROW
 73 002E 89             MOVL R1,A       ;SAVE NEW VALUE
 74 002F 0314           ADD A,#L(-L(STLIN))     ;CHECK FOR END OF DISPLAY
 75 0031 E2             MOV A,HACC      ;SUBTRACT STLIN FROM A
 76 0032 03F8           ADD A,#L(-H(STLIN) - 1) ;CARRY WILL BE SET IF A WAS > OR =
 77 0034 E639           JNC UPTRU               ;NEW VALUE OK, LOAD SROW AND RETURN
 78 0036 27             CLR A      ;RESET SROW TO BEGINNING
 79 0037 C2             MOV HACC,A
 80 0038 89             MOVL R1,A
 81 0039 99      UPTRU:  MOVL A,R1       ;LOAD R1 INTO SROW
 82 003A OE             MOV SROW,A
 83 003B 93             RETR
 84
 85 003C 99      DOWN:   MOVL A,R1       ;SCROLL DOWN
 86 003D 03F3           ADD A,#-13      ;SUBTRACT TO NEXT ROW
 87 003F AA             MOV R2,A        ;TEMP SAVE OF LOW ORDER
 88 0040 E2             MOV A,HACC      ;NOW DO UPPER HALF
 89 0041 03FF           ADD A,#OFF      ;CARRY WILL BE SET IF A WAS 12 OR MORE
 90 0043 F64D           JC DNTRU        ;NEW VALUE OK, LOAD VALUE INTO SROW
 91 0045 2307           MOV A,#H(LSTLIN)        ;RESET SROW TO LAST ROW
 92 0047 C2             MOV HACC,A
 93 0048 23DF           MOV A, #L(LSTLIN)
 94 004A 89             MOVL R1,A
 95 004B OE             MOV SROW,A
 96 004C 93             RETR
 97 004D C2      DNTRU:  MOV HACC,A
 98 004E FA             MOV A,R2
 99 004F 89             MOVL R1,A
100 0050 OE             MOV SROW,A
101 0051 93             RETR
102             .FORM
103
104             ;START OF INITIALIZING CODE
105
106 0052 C5     BEGIN:  SEL MBO
107 0053 C3             SEL RBO
108 0054 15             DIS XI    ;INTERRUPTS OFF FOR NOW
109 0055 35             DIS II
110 0056 65             STOP T    ;TIMER OFF
111 0057 27             CLR A      ;SET UP TIMING CHAIN FOR DEMO BOARD
112 0058 87             MOV TCP, A
113 0059 2367           MOV A, #103     ;HORIZONTAL LENGTH
114 005B B7             MOV @TCP, A
115 005C 230C           MOV A, #12      ;CHARACTERS/ROW
116 005E B7             MOV @TCP, A
117 005F 2353           MOV A, #83      ;HORIZONTAL SYNC BEGIN
118 0061 B7             MOV @TCP, A
119 0062 2363           MOV A, #99      ;HORIZONTAL SYNC END
120 0064 B7             MOV @TCP, A
121 0065 2391           MOV A, #091     ;CHARACTER HEIGHT/ EXTRA SCANS
122 0067 B7             MOV @TCP, A
123 0068 231A           MOV A, #26      ;VERTICAL LENGTH
124 006A B7             MOV @TCP, A
125 006B 2318           MOV A, #24      ;VERTICAL BLANK
126 006D B7             MOV @TCP, A
127 006E 2362           MOV A, #062     ;VERTICAL SYNC BEGIN/END
128 0070 B7             MOV @TCP, A
129 0071 2300           MOV A, #00      ;STATUS ROW BEGIN
130 0073 B7             MOV @TCP, A
131 0074 23F4           MOV A, #0F4     ;BLINK RATE
132 0076 B7             MOV @TCP, A
133 0077 2330           MOV A, #030     ;GRAPHICS COLUMN REGISTER
134 0079 B7             MOV @TCP, A
135 007A 2336           MOV A, #036     ;GRAPHICS ROW REGISTER
136 007C B7             MOV @TCP, A
137 007D 2389           MOV A, #089     ;UNDERLINE SIZE REGISTER
138 007F B7             MOV @TCP, A
139 0080 2309           MOV A, #009     ;CURSOR SIZE REGISTER
140 0082 B7             MOV @TCP,A
141
142
```

TL/DD/6970-7

```
143                     .FORM
144 0083 2304                   MOV A, #004     ;SET SYSTEM CONTROL REGISTER
145 0085 55                     MOV SCR, A      ;8 BI, 7 DOTS, DIVIDE 1, SEQUENTIAL LOOKUP
146
147 0086 27                     CLR A       ;SET VIDEO RAM POINTERS
148 0087 C2                     MOV HACC, A     ;ACCUMULATOR CLEARED
149 0088 0D                     MOV BEGD, A
150 0089 0E                     MOV SROW, A     ;SROW WILL BE OUT HOME
151 008A 88                     MOVL RO, A      ;CLEAR MEMORY POINTER
152 008B 89                     MOVL R1, A      ;1 IS SROW IMAGE
153 008C 2307                   MOV A, #H(MEMEND +1)
154 008E C2                     MOV HACC, A
155 008F 23EC                   MOV A,#L(MEMEND +1)
156 0091 0C                     MOV ENDD,A      ;SET END OF MEMORY POINTER
157 0092 8A                     MOV HOME, A     ;SET POINTER TO TITLE ROW
158 0093 23FF                   MOV A, #OFF
159 0095 C2                     MOV HACC,A
160 0096 8B                     MOV CURS, A     ;NO CURSOR
161 0097 3C                     MOV ALO, A      ;NO ATTRIBUTES FOR LATCH O
162 0098 3D                     MOV AL1, A      ;NO ATTRIBUTES FOR LATCH 1
163 0099 85                     CLR FO    ;FO IS "START" BIT
164 009A A5                     CLR F1    ;F1 IS "THROUGH" BIT
165 009B 2320                   MOV A,#020
166 009D 82                     MOV MASK,A      ;SET INTERRUPT MASK
167
168 009E 3456                   CALL MEMCLR     ;CLEAR VIDEO MEMORY
169
170 00A0 05                     EN XI       ;REENABLE EXTERNAL INTERRUPTS
171 00A1 2400                   JMP LINNUM      ;DISPLAY TITLE INFORMATION
172
173 00A3 86A3   KEYIN:  JNFO KEYIN      ;WAIT FOR KEYBOARD INPUT
174
175                     .FORM
176
177                     ;DATA ACQUISITION ROUTINES
178
179 00A5 85              CLR FO    ;CLEAR START BIT
180 00A6 2300           MOV A,#VOFF     ;VIDEO OFF
181 00A8 45             MOV VCR,A
182 00A9 3456           CALL MEMCLR     ;CLEAR VIDEO MEMORY
183 00AB 27             CLR A
184 00AC C2             MOV HACC,A
185 00AD 0E             MOV SROW,A      ;RESET SROW TO BEGINNING
186 00AE BA9C           MOV R2, #156    ;SET LOOP COUNTER FOR # WORDS TO READ
187 00B0 2305           MOV A,#5
188 00B2 88             MOVL RO,A       ;SET MEMORY POINTER TO FIRST DATA DEPOSIT
189 00B3 23C0           MOV A,#0C0
190 00B5 C2             MOV HACC,A
191 00B6 89             MOVL R1,A       ;LOAD R1 WITH ADDRESS OF SWITCH REGISTER
192 00B7 BB0D           MOV R3,#13      ;LOAD R3 WITH POINTER BUMP CONSTANT
193 00B9 8C             MOV A,INTR      ;CLEAR OUT ANY PENDING INTERRUPTS
194 00BA 25             EN II       ;REENABLE LP INTERRUPT
195
196 00BB 66BB   DATAIN: JNF1 DATAIN     ;WAIT FOR "THROUGH" BIT TO SET
197
198                     ;DISASSEMBLE DATA INTO DISPLAY FORMAT
199 00BD A5             CLR F1    ;RESET "THROUGH"
200 00BE BA9C           MOV R2,#156     ;DISASSEMBLE DATA, LOAD WORD COUNTER
201 00C0 27             CLR A
202 00C1 C2             MOV HACC,A
203 00C2 2305           MOV A,#5
204 00C4 88             MOVL RO,A       ;SET MEMORY POINTER TO FIRST DATA DEPOSIT
205 00C5 BB08           MOV R3,#8       ;DO 8 BITS
206 00C7 90     UNASS:  MOVX A,@RO      ;LOAD IN DATA BYTE
207 00C8 04CB           JMP DEPST
208 00CA FC     RETRIV: MOV A,R4        ;RETRIEVE CHARACTER
209 00CB F7     DEPST:  RLC A     ;ROTATE BIT INTO CARRY
210 00CC AC             MOV R4,A        ;TEMPORARY SAVE
211 00CD F6D3           JC ONE    ;STORE A "1"
212 00CF 2330           MOV A,#'0'      ;STORE A "0"
213 00D1 04D5           JMP CONT
```

TL/DD/6970-8

7

```
214 00D3 2331    ONE:    MOV A,#'1'      ;STORE A "1"
215 00D5 80      CONT:   MOVX @RO,A      ;STORE CHARACTER
216 00D6 38              INCL RO  ;INCREMENT POINTER
217 00D7 EBCA            DJNZ R3,RETRIV  ;CONTINUE IF WORD NOT DONE
218
219 00D9 98              MOVL A,RO       ;BUMP MEMORY POINTER TO NEXT WORD
220 00DA 0305            ADD A,#5
221 00DC 88              MOVL RO,A
222 00DD BB08            MOV R3,#8       ;RESET BIT COUNTER
223 00DF EAC7            DJNZ R2,UNASS   ;CONTINUE IF ALL LOCATIONS NOT DONE
224 00E1 2400            JMP LINNUM      ;JUMP TO NEXT PAGE
225              .FORM
226              ;LINE NUMBERING ROUTINES
227      0100    . = 0100
228
229 0100 27      LINNUM: CLR A
230 0101 C2              MOV HACC,A
231 0102 88              MOVL RO,A       ;CLEAR MEMORY POINTER
232 0103 BA9C            MOV R2,#156     ;DO 156 LINES
233 0105 BB0A            MOV R3,#10      ;SET HUNDREDS POINTER
234 0107 BC0A            MOV R4,#10      ;SET TENS POINTER
235 0109 BD0A            MOV R5,#10      ;SET ONES POINTER
236
237 010B FB      NUMLP:  MOV A,R3        ;LOOK UP HUNDREDS ASCII CODE
238 010C 343B            CALL LKUP
239 010E 80              MOVX @RO,A      ;STORE HUNDREDS ASCII CODE
240 010F 38              INCL RO
241 0110 FC              MOV A,R4        ;LOOK UP TENS ASCII CODE
242 0111 343B            CALL LKUP
243 0113 80              MOVX @RO,A      ;STORE TENS ASCII CODE
244 0114 38              INCL RO
245 0115 FD              MOV A,R5        ;LOOK UP ONES ASCII CODE
246 0116 343B            CALL LKUP
247 0118 80              MOVX @RO,A      ;STORE ONES ASCII CODE
248
249 0119 98              MOVL A,RO       ;BUMP RO TO NEXT LINE
250 011A 030B            ADD A,#11
251 011C 88              MOVL RO,A
252
253 011D ED26            DJNZ R5,CONNUM  ;INCREMENT ONES POINTER
254 011F BD0A            MOV R5,#10      ;MUST NOW INCREMENT TENS
255 0121 EC26            DJNZ R4,CONNUM  ;INCREMENT TENS POINTER
256 0123 BC0A            MOV R4,#10      ;MUST NOW INCREMENT HUNDREDS
257 0125 CB              DEC R3
258
259 0126 EA0B    CONNUM: DJNZ R2,NUMLP   ;DO ANOTHER ROW
260
261 0128 9A              MOV A,HOME      ;PUT UP TITLE LINE
262 0129 88              MOVL RO,A
263 012A BA0D            MOV R2,#13      ;LOOKUP 13 CHARACTERS
264 012C BB49            MOV R3,#L(TITLE)        ;LOAD POINTER TO ASCII TITLE STRING
265
266 012E FB      TITLP:  MOV A,R3
267 012F B3              MOVP A,@A
268 0130 80              MOVX @RO,A
269 0131 38              INCL RO
270 0132 1B              INC R3
271 0133 EA2E            DJNZ R2,TITLP
272
273              .FORM
274
275 0135 2320            MOV A,#VON      ;TURN VIDEO BACK ON
276 0137 45              MOV VCR,A
277 0138 05              EN XI      ;REENABLE KEYBOARD INTERRUPTS
278 0139 04A3            JMP KEYIN      ;RETURN AND WAIT FOR NEXT START
279
280              ;SUBROUTINES
281
282 013B 033E    LKUP:   ADD A,#L(CHAR)-1;
283 013D B3              MOVP A,@A       ;LOOK UP ASCII NUMBER
284 013E 93              RETR        ;RETURN
```

TL/DD/6970–9

```
285
286 013F 39       CHAR:   .BYTE '9876543210'
287
288 0149 53       TITLE:  .BYTE 'STEP 7 DATA O'
289
290 0156 27       MEMCLR: CLR A    ;VIDEO MEMORY CLEAR LOOP
291 0157 C2               MOV HACC,A       ;ACCUMULATOR CLEAR
292 0158 88               MOVL RO,A        ;RO CLEAR
293 0159 BA00             MOV R2,#0        ;INNER LOOP COUNTER SET
294 015B BB08             MOV R3, #8       ;OUTER LOOP COUNTER SET
295 015D 2320             MOV A,#020       ;SPACE CHARACTER
296 015F 80       MCLRLP: MOVX @RO,A       ;STORE A CHARACTER
297 0160 38               INCL RO  ;INCREMENT POINTER
298 0161 EA5F             DJNZ R2,MCLRLP   ;TEST INNER LOOP
299 0163 BA00             MOV R2,#0        ;RELOAD INNER LOOP COUNTER
300 0165 EB5F             DJNZ R3, MCLRLP ;TEST OUTER LOOP
301 0167 93               RETR         ;THROUGH, RETURN
302
303               .END
BEGIN   0052    CHAR    013F    CKOFF   0024    CONNUM  0126
CONT    00D5    DATAIN  00BB    DEPST   00CB    DNTRU   004D
DOWN    003C    EXI     0003 *  INI     0007 *  KEY     0012
KEYIN   00A3    LINNUM  0100    LKUP    013B    LSTLIN  07DF
MCLRLP  015F    MEMCLR  0156    MEMEND  07EB    NOTRU   0011
NUMLP   010B    ONE     00D3    RESET   0000 *  RETRIV  00CA
START   0027    STLIN   07EC    TITLE   0149    TITLP   012E
UNASS   00C7    UP      002B    UPTRU   0039    VOFF    0000
VON     0020

NO ERROR LINES

  328  ROM BYTES USED

SOURCE CHECKSUM = 40D8
OBJECT CHECKSUM = 0649

INPUT   FILE A:LOGIC.MAC
LISTING FILE A:LOGIC.PRN
OBJECT  FILE A:LOGIC.LM
```

TL/DD/6970–10

Building an Inexpensive but Powerful Color Terminal

National Semiconductor
Application Note 374
Leigh Cropper

Historically, the design of a color CRT terminal has involved a significant upgrade of the circuit for a monochrome terminal. The result was a stiff increase in price for the electronics as well as for the monitor when going from monochrome to color. As a result, most companies built monochrome terminals and a few built color terminals only.

On the personal computer front where separate monitors are common, manufacturers have started to offer video cards which will support either a monochrome or a color monitor. More recently, color terminals have begun to appear which are extensions of monochrome terminal families. They require a board full of I.C.s for even the most space efficient designs. But now, using the TMP, you can do the same job with just one VLSI chip and a half dozen 7400 family TTL chips.

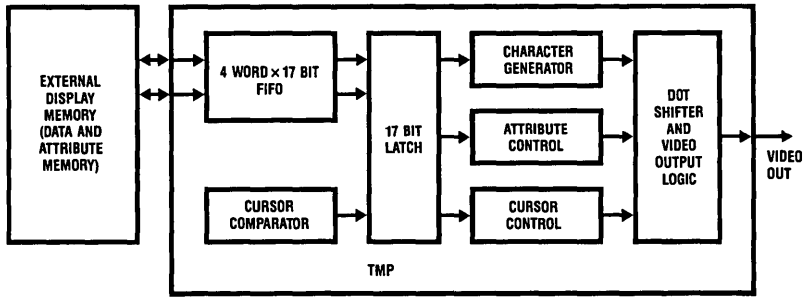
The National Semiconductor NS405 Series Terminal Management Processor (TMP) was originally conceived as a monochrome "terminal on a chip". However, the design team took special pains to build in "hooks" to allow users to augment the basic features of the TMP. In particular the TMP supports almost unlimited attribute expansion, and therein lies the key to adding color to a TMP-based terminal. Even nicer, the addition of color attributes does not sacrifice any of the other powerful features provided by the TMP.

Here, we will delve into a little of the mechanics of TMP attribute handling. The diagram in *Figure 1* shows the path of the attribute bits (normally 8) from the display memory into the TMP, through the FIFO, the attribute control logic, and finally to the video output section where the attributes are combined with the serialized video output.

Because the display memory space may be large (up to 64k x 16), it is easy to store many more attribute bits by adding display memory chips. A 2k x 8 RAM will hold 8 attribute bits for every location on an 80 row by 25 line display. However, in order to implement color attributes, three problems must be examined: (1) how to let both the CPU and the display controller address the extra attribute memory in a practical manner; (2) how to imitate the behavior of the internal FIFO and maintain proper synchronization; and (3) how to combine the color attributes and the video output signal.

Before addressing the three problems in detail, a discussion of the number and type of color attributes is in order. The simplest type of color display would require only 3 bits (red, green, and blue). That allows a character to be displayed in any of 7 colors over a black background or, when reverse video is asserted, the character is black on a colored background. For independent control of both the foreground and background colors, 6 bits are required. To get more shades of color, add more bits.

A practical approach employs a 2k x 8 RAM for the color attribute memory. Three of the bits control the foreground color, three control the background color and the remaining two may be used to adjust intensity (1 for foreground and 1 for background).



TL/DD/7923-1

FIGURE 1. TMP Attribute Processing

Now let's tackle the problems one by one.

1. COLOR ATTRIBUTE MEMORY ADDRESSING. When fetching data for the display, we need to get 24 bits in parallel (8 data, 8 attribute and 8 color attribute). But when the CPU accesses memory, it can handle only 16 bits at a time, so the CPU must be able to read and write color attributes in a different bank of memory from that where the data and ordinary attributes are stored. For an 80 character by 25 row display the memory could be mapped as shown in *Figures 2* and *3:*

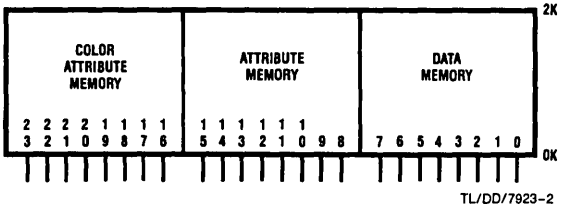


FIGURE 2. Memory Map as Selected for Screen Refresh

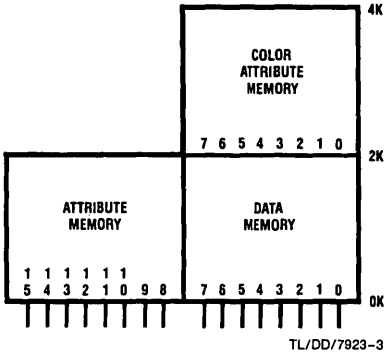


FIGURE 3. Memory Map as Selected for CPU Access

The mapping is implemented by the following circuit:

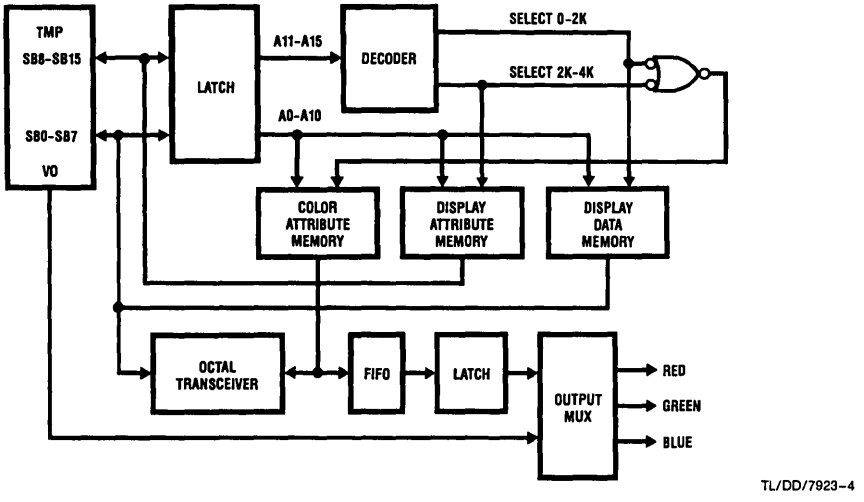


FIGURE 4. Color Attribute Memory Mapping Circuit

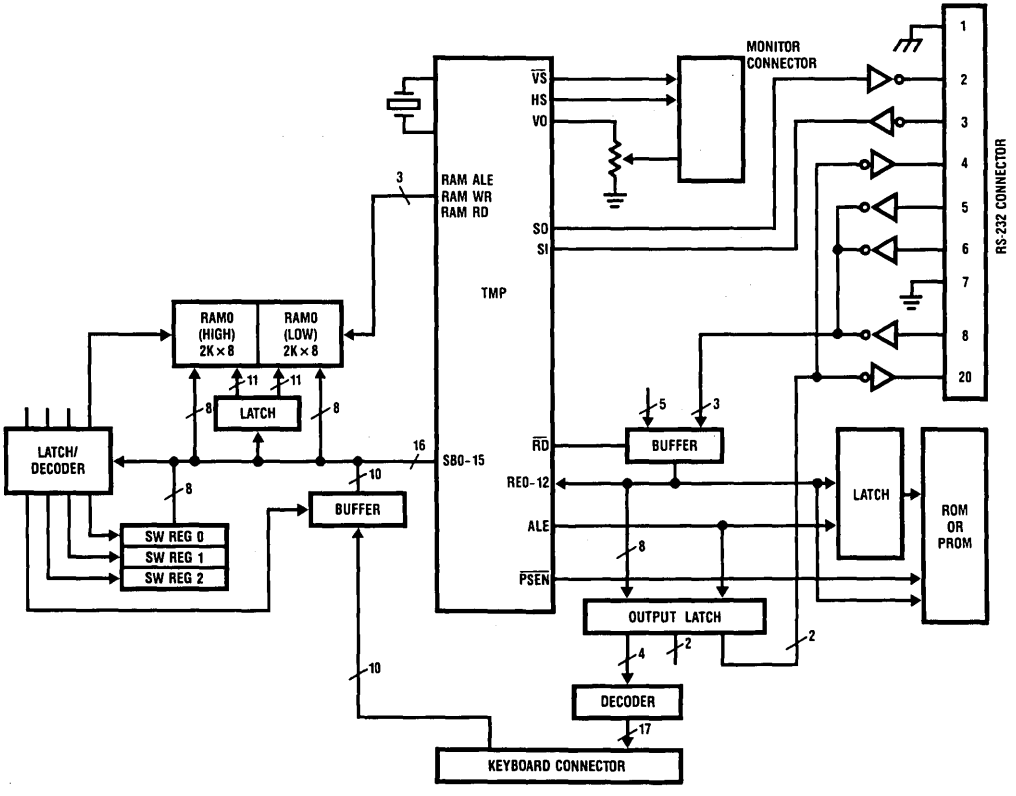
During a display refresh cycle the color attribute memory is selected by the low bank select (the same select signal that enables the data and attribute memories). However, the color attribute bits drive the external FIFO's, whereas the output from the other two memories is routed through the TMP. The data path from the color attribute memory to the TMP is buffered by an octal transceiver which is disabled when the low bank is selected. During a CPU access to color attribute memory, the high bank select enables the color attribute memory and the octal transceiver. The direction control of the transceiver is controlled by the RAM RD signal from the TMP.

2. EXTERNAL FIFO SYNCHRONIZATION. The TMP provides FI CLK (FIFO Input Clock) and FO CLK (FIFO Output clock) signals which may be used to clock an external FIFO. The FI CLK signal is identical in timing and duty cycle to the RAM RD signal except that FI CLK is disabled (stays high) when the CPU accesses display memory. When the 74LS224 is used as an external FIFO, FI CLK must be inverted. The rising edge of FO CLK occurs when output of the internal FIFO is loaded into the internal dot shifter. The FO CLK is used to empty a word from the external FIFO and clock it into an octal latch.
3. COMBINING COLOR ATTRIBUTES WITH VIDEO. When using foreground and background color attributes, a 74LS157 multiplexer works nicely to switch between the two. The eight color attribute bits from the latch are separated into groups of four. The video output signal is used to switch the multiplexer. When the video output is high, foreground attributes are selected: and when the video output is low, background attributes are selected. The outputs of the multiplexer (red, green, blue and intensity) directly drive the color monitor inputs. A minor problem arises because the video output from the TMP already includes the blanking signal. That makes it impossible to differentiate between a series of spaces in the middle of the screen and the horizontal blanking interval. In either case, the video output is low. The easiest solution is handled in software. Let's assume that we want an 80 column display and are using three 2K x 8 memory chips for the data, attribute and color memories. We set up the TMP for 81 columns and then configure the program so that the 81st column always contains a space code with all attribute bits off (including color). That way the background color will always be black during both horizontal and vertical retrace. The cost is 25 locations in each of the memories, but we can afford that many because an 80 x 25 display requires 2000 locations, leaving 48 free.

A Practical Example

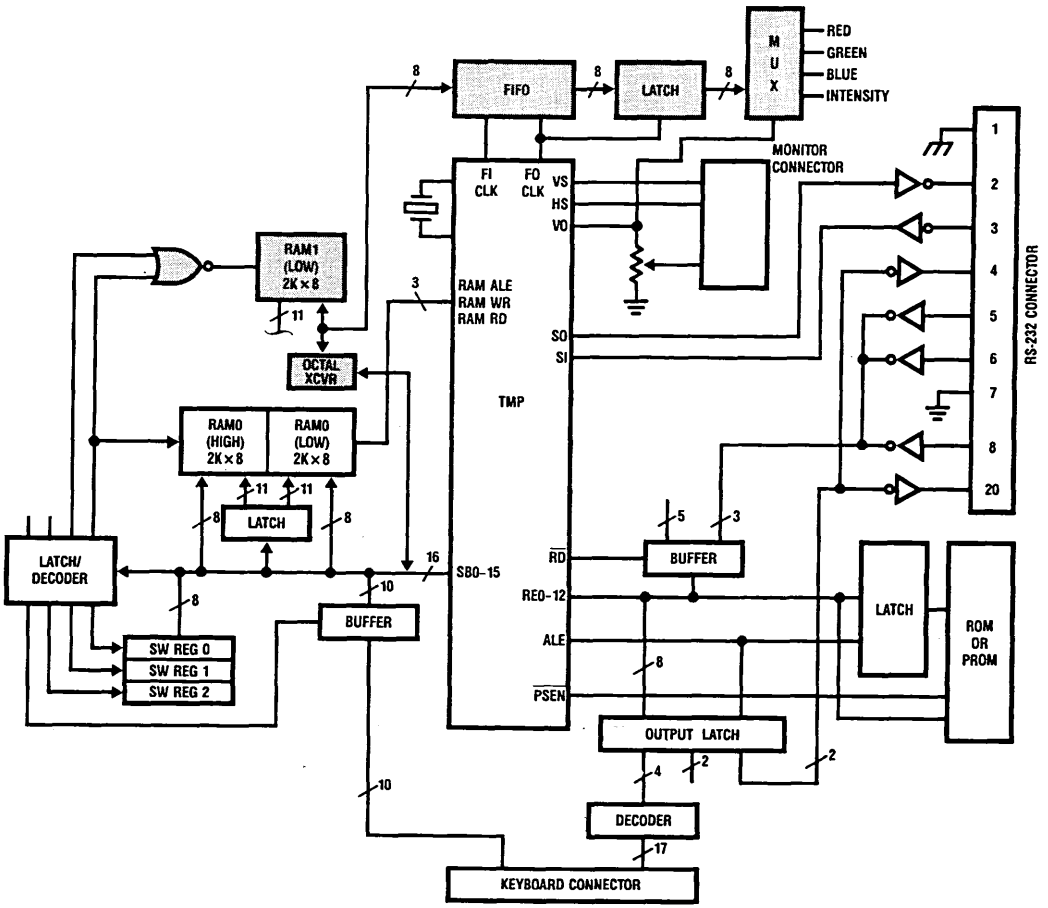
Here we will present a color terminal circuit with the associated program as an example of what you may want to do. We started with the terminal design of the TMP development board. See the block diagram in *Figure 5*.

The block diagram of the color terminal (with the old portions of the original monochrome terminal unshaded and the new color circuits shaded) appears in *Figure 6*. The new circuitry was added in the prototyping area of the development board.



TL/DD/7923-5

FIGURE 5. TMP Development Board Block Diagram



TL/DD/7923–6

FIGURE 6. TMP Development Board Color Circuitry Block Diagram

COLOR ATTRIBUTE BIT ASSIGNMENTS

The bit assignments are:

Bit 0 — Blue foreground
Bit 1 — Green foreground
Bit 2 — Red foreground
Bit 3 — Blue background
Bit 4 — Green background
Bit 5 — Red background
Bit 6 — Foreground intensity
Bit 7 — Background intensity

Without using the intensity control bits you get 8 foreground colors: red, green, blue, magenta, cyan, yellow, white, and black (beam off). The same 8 colors may be independently selected for the background. There are several RGB monitors available in the moderate price range with sufficient bandwidth to work with a 12 MHz TMP. Some of them include a separate intensity (or luminence) input. Others include internal decoding circuitry which provides the ability to handle 4 bits of color input and provide as many as 16 different colors.

The demonstration program which runs on the development board allows limited color support. The Escape, V sequence from the keyboard or the receiver prompts the program to treat the next character received as an eight bit color attribute byte with the bit assignments as listed above. That byte is written to the color attribute memory as each succeeding character is received, until another escape, V sequence is encountered. The table which follows includes the foreground and background color combinations for characters which can be entered from the keyboard, but it ignores the effect of the 2 high-order bits (foreground and background intensity).

COLOR COMBINATIONS FOR RGB MONITORS

Table I gives the Foreground/Background color combinations that occur when using the '<ESC> Vv' Escape sequence. To set the current color attribute, all that you need to do is select the color combination from the Table below, and send it to the NS405 as part of the <ESC> Vx sequence. For example, '<ESC> V"' causes the Foreground color to be green and the Background color to be red . . . not all that pleasing, to my tastes, but choose what you will.

TABLE I
Foreground/Background Color Combinations

Char	Fore/Back	Char	Fore/Back	Char	Fore/Back
sp	Black/Red	6	Yellow/Yellow	K	Cyan/Blue
!	Blue/Red	7	White/Yellow	L	Red/Blue
"	Green/Red	8	Black/White	M	Magenta/Blue
#	Cyan/Red	9	Blue/White	N	Yellow/Blue
$	Red/Red	:	Green/White	O	White/Blue
%	Magenta/Red	;	Cyan/White	P	Black/Green
&	Yellow/Red	<	Red/White	Q	Blue/Green
'	White/Red	=	Magenta/White	R	Green/Green
(	Black/Magenta	>	Yellow/White	S	Cyan/Green
)	Blue/Magenta	?	White/White	T	Red/Green
*	Green/Magenta	@	Black/Black	U	Magenta/Green
+	Cyan/Magenta	A	Blue/Black	V	Yellow/Green
,	Red/Magenta	B	Green/Black	W	White/Green
-	Magenta/Magenta	C	Cyan/Black	X	Black/Cyan
.	Yellow/Magenta	D	Red/Black	Y	Blue/Cyan
/	White/Magenta	E	Magenta/Black	Z	Green/Cyan
0	Black/Yellow	F	Yellow/Black	[	Cyan/Cyan
1	Blue/Yellow	G	White/Blue	\	Red/Cyan
2	Green/Yellow	H	Black/Blue	]	Magenta/Cyan
3	Cyan/Yellow	I	Blue/Blue	^	Yellow/Cyan
4	Red/Yellow	J	Green/Blue	-	White/Cyan
5	Magenta/Yellow				

TMP Extended Program Memory Application Note

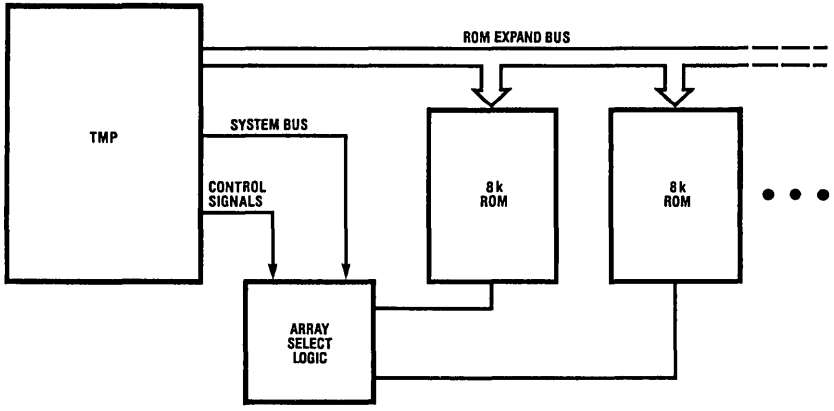
National Semiconductor
Application Note 399
Richard Lazovick

OVERVIEW/INTRODUCTION

The purpose of this application note is to describe methods for expanding the program memory of the NS405 series TERMINAL MANAGEMENT PROCESSOR (TMP) and to provide direction in software techniques for utilizing the expanded memory efficiently. The chip has a built-in capability of addressing up to 8k of external program memory (ROM), via the ROM Expand Bus, and 64k of video display memory (RAM), via the System Bus. Although 8k of program memory is sufficient for most applications there are many applications, such as emulating multiple terminals or using many look-up tables, that require still more memory. However, it is very rare that the entire 64k of video RAM is used since that is more than enough memory to store two screens of data in the pixel mode or thirty-two screens of data in the alphanumeric/block graphics mode. Therefore it is practical to use a video memory address to switch between two or more 8k memory arrays.

The idea behind using a bank select switch to change from one memory array to another is not new, nor is it difficult, and when implemented properly it can be a very useful tool. The TMP has all the necessary control signals to make both the software and hardware straight-forward.

Block Diagram



TL/DD/8430–1

SOFTWARE

For purposes of demonstration it will be easier to look at the software aspects of using an array select switch first, then designing the hardware to implement it.

The easiest case occurs if we use less than 16k of display memory. Then we have two system bus address lines available to select either of our two arrays. To switch arrays all we have to do is read from (or write to) an address that uses the address line you wish to toggle. It is safer to read from the address since we do not want to change data in memory at the location addressed by the lower order address lines.

Suppose we choose SB14 to select the low order array and SB15 to select the high order array. The program steps we would go through to switch from the low array to the high array could be:

```
MOV   A,#080        ;Load HACC w/ 80H to set SB15 HI
MOV   HACC,A        ;and set SB14 LO. We do not care
MOVL  R0,A          ;about the other address bits.
MOVX  A,@R0         ;SB15 goes HI.
```

In general we will want to switch arrays several times, and we will want to be able to conveniently control the destination address in the new array.

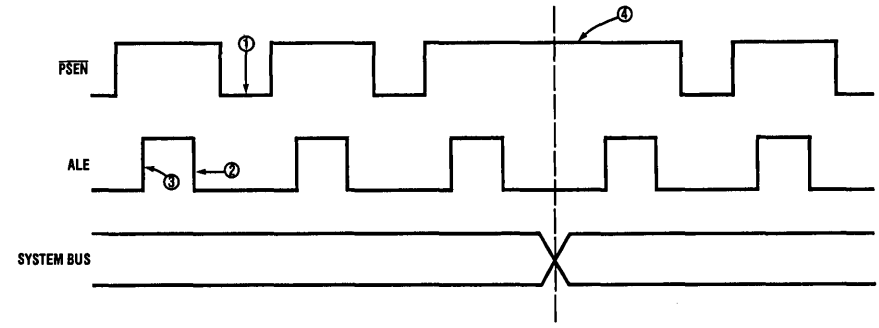
Since it is very cumbersome to rewrite the whole sequence everytime, let's mimic the internal select memory bank command (SEL MBx) by using a subroutine and a CALL followed by a JMP to conveniently control our array switching.

```
        CALL  SELHA       ;Select HI order array.
        JMP   HERE        ;Jump to HERE in new array.

SELHA:  MOV   A,#080      ;Load HACC w/ 80H to set SB15 HI
        MOV   HACC,A      ;and set SB14 LO. We do not care
        MOVL  R0,A        ;about the other address bits.
        MOVX  A,@R0       ;SB15 goes HI.
        RET               ;Return to execute the jump.
```

Now each time we switch to the high order array all we have to do is execute a CALL and a JMP.

System Signals Timing Diagram



Note 1: Enable ROM output drivers.

TL/DD/8430-2

Note 2: ROM address available.

Note 3: RE bus addresses changes during rising edge of ALE and are stable by falling edge.

Note 4: No $\overline{\text{PSEN}}$ signal present during last cycle of MOVX instruction, however $\overline{\text{PSEN}}$ is active during both RET cycles.

HARDWARE

Now that we have made the software simple and straightforward we have to look at what hardware is necessary to implement it.

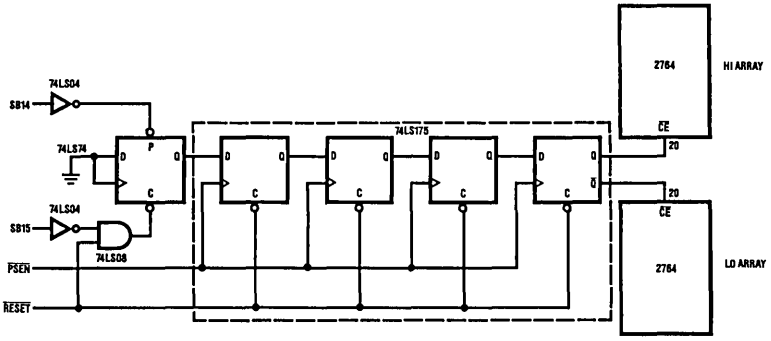
We want to: 1) create two mutually exclusive enable signals—one for each array,
2) be able to easily use and latch the address line signals, and
3) delay the actual switching of arrays until after the jump instruction, with the new address, is read into the TMP from the old array.

Looking at the program we see that the system bus changes after the MOVX instruction with the RET and JMP instructions still to be read in from memory before we actually want to switch arrays. Each instruction takes two cycles, therefore we want to delay our array switching signal by four cycles. Looking at all the output signals on the TMP there are two possible signals to use as a clock to delay the array switching signal. These signals are $\overline{PSEN}$ and ALE (see System Signals Timing Diagram).

The main disadvantage of using ALE is that whereas we want a rising edge to clock the flip-flops used for the delay, the ROM addresses are not stable until the falling edge of ALE. Therefore, we save one inverter by using $\overline{PSEN}$.

One possible circuit implementation is shown below:

Circuit Diagram



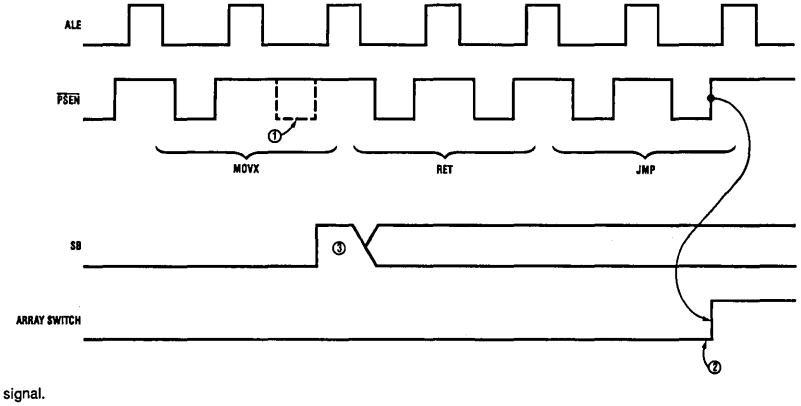
TL/DD/8430-3

The first flip-flop latches one of the two system bus signals and the next four delay the array switching signal by four $\overline{PSEN}$ cycles. The two inverters are there so that we trigger off a ONE on the address. If the system bus was configured as a 16 bit address/data bus (bit 4 of SCR set) then the latched address lines would have to be used. Since it is always desirable to have the flip-flops in a known state at power up, some sort of reset circuitry should be used (e.g., by tying the power up reset circuit to the clear inputs on the flip-flops), or both arrays should have identical reset sequences that include setting the flip-flops to a known state.

LOOKING IN DEPTH

Now that we have the basic software format we are going to use and the hardware to implement it, let's look at what is actually happening (see Array Switching Timing Diagram 1). The system bus line switches after the first cycle of the MOVX instruction, but there is no $\overline{PSEN}$ during the second cycle. Then there are two $\overline{PSEN}$ signals during the RET instruction and two $\overline{PSEN}$ signals during the JMP instruction. Just after the second byte of the JMP is read into the TMP, the arrays are switched, the PC gets loaded with the new address, and the program continues execution as normal in the new array from the address indicated in the JMP instruction. Be sure you understand the Array Switching Timing Diagram 1 before continuing.

Array Switching Timing Diagram 1



TL/DD/8430-4

Note 1: No $\overline{PSEN}$ signal.

Note 2: Arrays switch here.

Note 3: Valid approximately 360 ns.

ADDENDUMS

Although it can be a problem when trying to execute a call across array boundaries, the problem can be easily overcome as can the confusion that arises when many array switchings occur. All that one needs to do is to organize the program memory efficiently. One such scheme would be to set aside a block of memory in each bank, such as the last page to use for memory mapping. For example if we wanted to jump from location HOME in the LO array to location HERE in the HI array and then back to HOME we could map our memory as shown below:

Lo Array

MAIN PROGRAM:

```
                .
                .
                .
0500  HOME:  JMP   HERE
                .
                .
                .
```

ARRAY SWITCHING SUBROUTINE:

```
0700  SELHA: MOV   A,#080
0702         MOV   HACC,A
0703         MOVL, R0,A
0704         MOVX  A,@R0
0705         NOP
0706         NOP
0707         RET
```

MEMORY MAP:

```
070A  HERE:  CALL  SELHA
070C         JMP   HOME
070E        (CALL SELHA)
0710        (JMP addr)
                .
                .
                .
```

Hi Array

```
                .
                .
                .
0680  HERE:  NOP
0681         JMP  HOME
                .
                .
                .
```

```
0700  SELLA: MOV  A,#040
0702         MOV  HACC,A
0703         MOVL R0,A
0704         MOVX A,R0
0705         NOP
0706         NOP
0707         RET
```

```
070A  HOME:  CALL SELLA
070C         JMP  HERE
070E        (CALL SELHA)
0710        (JMP addr)
                .
                .
                .
```

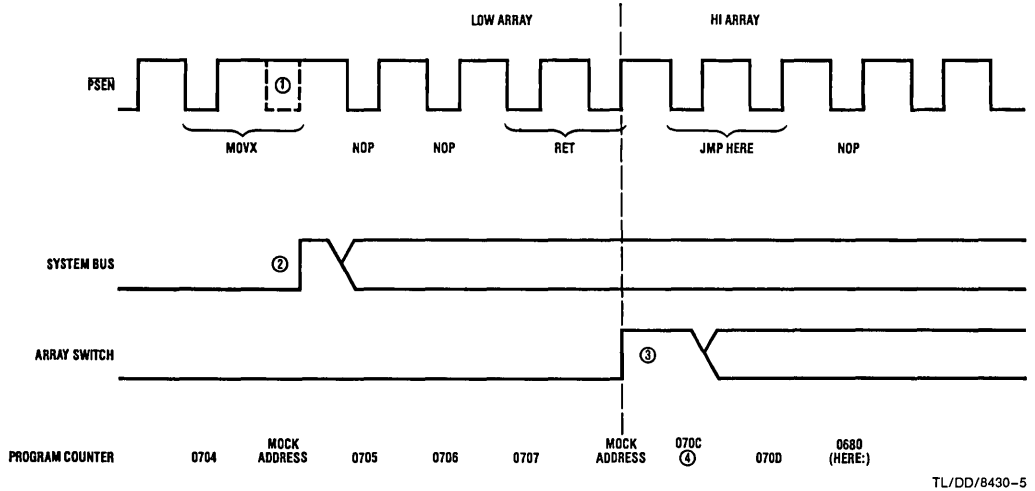
TL/DD/8430-6

Note: The arrows show which JMP corresponds to which subroutine call. For example the JMP HOME at location 070C in the LO ARRAY corresponds to the CALL at location 070A in the HI ARRAY. The ()'s show how the CALL's and JMP's can be strung together in a neat pattern.

Notice that there are two NOP's in the subroutine. Since the JMP after the CALL was moved to the new array the two NOP's were added to the subroutine so that the actual array switching occurs just after the completion of the RET instruction. The PC is then loaded with the new jump value, loaded in from the new array, and we continue execution as expected. Be sure to understand the timing before going any further (see Array Switching Timing Diagram 2). The way the memory map is set up it is easy to organize and keep track of jumps using the pattern indicated in the example. This method also eliminates any problems with the assembler searching for undefined labels.

ADDENDUMS (Continued)

Array Switching Timing Diagram 2



TL/DD/8430–5

Note 1: Missing $\overline{\text{PSEN}}$ signal.
Note 2: System bus changes.
Note 3: Arrays switch.
Note 4: Now in new array.

Since there are two extra NOP's in the switching array subroutine the hardware can now be simplified and the system speed increased by removing two of the flip-flops from the chain. Through the use of the SEL MBx commands the memory map can be located in any page of any memory bank. For example if we wanted to jump to location HERE in memory bank 2 of the HI array from memory bank 1 of the LO array (after having removed two flip-flops) we could map our memory as shown below:

Lo Array

ARRAY SWITCHING SUBROUTINE:

```
0700  SELHA:  MOV   A,#080
0702          MOV   HACC,A
0703          MOVL  R0,A
0704          MOVX  A,@R0
0705          RET
```

MEMORY MAP:

```
070A  HERE:   CALL  SELHA
070C
070E
```

MAIN PROGRAM:

```
0800  HOME:   SEL   MB0
0801          JMP   HERE
```

Hi Array

```
0700  SELLA:  MOV   A,#040
0702          MOV   HACC,A
0703          MOVL  R0,A
0704          MOVX  A,@R0
0705          RET

070A
070C          SEL   MB2
070E          JMP   HERE

1000  HERE:   NOP
```

If a call into the other array is necessary a similar pattern to that above could be used. Start by replacing the JMP's with CALL's to the desired subroutine and appropriately placing returns. For example (here it comes) if we wanted to CALL HOME from HERE we could memory map as shown below.

Lo Array	Hi Array

MAIN PROGRAM:

```
HOME:  NOP
       RET
```

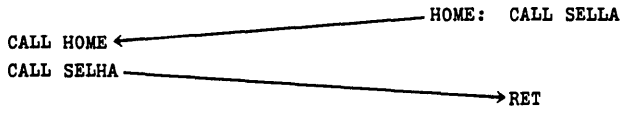
```
HERE:  NOP
       CALL HOME
```

ARRAY SWITCHING SUBROUTINE:

```
SELHA:  .
        .
        .
        RET
```

```
SELLA:  .
        .
        .
        RET
```

MEMORY MAP:



TL/DD/8430–7

Since calling between different memory banks is not straight forward it is advisable to be very careful when doing it, or to limit calls between arrays only to those that reside in the same memory bank.

HELPFUL HINTS

These schemes can all be modified to multiple arrays and easier or fancier mappings, however there are a few things to keep in mind.

1) If using a system bus address line to toggle the array, don't use that line as part of an actual display memory address.
2) The MOVX instruction can require more than two cycles depending on system bus contention, however we are only concerned with the last two cycles and the $\overline{\text{PSEN}}$ signals that occur after the system bus line changes.
3) If using interrupts—disable them while switching arrays and keep all time critical routines in the same array.
4) A demux or decoder can be used to select memory arrays or decode address lines when more than two 8k arrays are implemented or more than 16k of video RAM is being used.
5) If extra memory is needed, but a good deal of the program memory is data storage, the data could be stored in the video memory space instead of implementing a new array.
6) If the TMP is going to be used in a noisy environment or the system bus is configured as a 16 bit bidirectional bus a synchronous latch should be used to assure stable levels on SB14 and SB15.
7) The given array switching circuit can be implemented with the demo board by wiring it into an extension board that can be plugged into the prom socket U9. Wire the two new proms in parallel with each other and with a cable that can plug directly into the prom socket. However, instead of using pin 20 from the demo board, use the two array enable lines as the chip enables for the 2764 proms.

   Also use SB12 and SB13 instead of SB14 and SB15.

Section 8

Microcontroller Development Support

Section 8 Contents

National Semiconductor

Development Support

Our job doesn't end when you buy a National microcontroller, it only begins.

The next step is to help you put that microcontroller to work—delivering real-world performance in a real-world application.

That's why we offer you such a comprehensive, powerful, easy-to-use package of development tools.

MICROCONTROLLER ON-LINE EMULATOR

Our Microcontroller On-Line Emulator (MOLE™) is a complete, inexpensive system designed to support both hardware and software development of all NSC microcontrollers.

Using standard computer platforms (IBM PC, VAX, and others), the MOLE system gives you the tools to write, assemble, debug, and emulate software for your target microcontroller, whether it belongs to the COP400 4-bit family, the COP800 8-bit family, or the HPC 16-bit family.

The MOLE system itself consists of two circuit boards that interface to each other and to the host computer using a MOLE software package.

One board is called the Brain Board and is common to all MOLE systems. It provides the major functional features of the system, linking the various elements, including other Brain Boards for a multi-workstation system tied to a single host.

The other board is called the Personality Board and is different for each microcontroller family. It provides the unique emulation functions for the system.

Your own computer CPU provides a powerful, cost-effective base for bulk storage of object code, disk editing and assembly, and for high-speed processing.

Using resident firmware in the Monitor section of each Personality Board, you can download results from your host computer, you can display and alter code in both hex and mnemonic format, you can set Breakpoints and Traces, you can execute Time measurements, and you can examine and modify internal registers and I/O.

Once you've got debugged code, you can transmit it directly to National, where we'll use it to create the tooling necessary for manufacturing the appropriate masks for your microcontroller.

DIAL-A-HELPER ON-LINE APPLICATIONS SUPPORT

Dial-A-Helper lets you communicate directly with the Microcontroller Applications Engineers at National.

Using standard computer communications software, you can dial into the automated Dial-A-Helper Information System 24 hours a day.

You can leave messages on the electronic bulletin board for the Applications Engineers, then retrieve their responses.

You can select and then download specific applications data.

And you can even arrange for the Applications Engineering Group to take over direct control of your MOLE system for particularly tough debug problems.

DIAL-A-HELPER

Voice: (408) 721-5582 (8 a.m.–5 p.m. PST)

Modem: (408) 739-1162 (24 Hrs./day)

Setup: Baud rate 300 bps or 1200 bps 8 bits, no parity, 1 stop

DEDICATED APPLICATIONS ENGINEERS

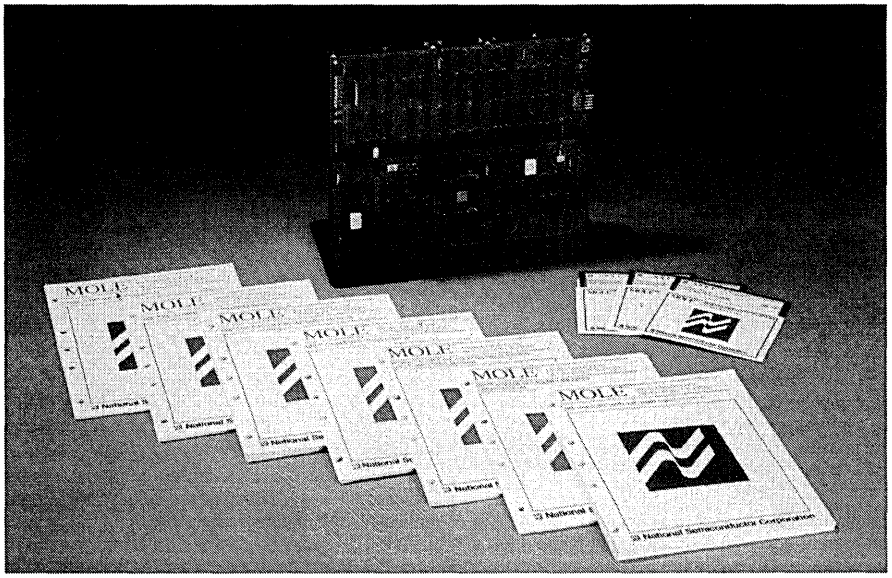
We've assembled a dedicated team of highly trained, highly experienced engineering professionals to help you implement your solution quickly, effectively, efficiently and to ensure that it's the best solution for your specific application.

At National, we believe that the best technology is also the most usable technology. That's why our microcontrollers provide such practical solutions to such real design problems. And that's why our microcontroller development support includes such comprehensive tools and such powerful engineering resources.

No one makes more microcontrollers than National and no one does more to help you put those microcontrollers to work.

Microcontroller Development Support

National Semiconductor
Application Note 456
Microcontroller Marketing



TL/DD/8830–14

MOLE™ DEVELOPMENT TOOLS

The MOLE (Microcontroller On Line Emulator) system is designed to support the development of NSC Microcontroller products. These include COPS™ family, and the HPC™ family of products. The MOLE provides effective support for the development of both software and hardware in Microcontroller-based applications.

The purpose of the MOLE is to provide the tools required to write and assemble code for the target microcontroller and assist in the debugging of both the hardware and software.

A MOLE system consists of three components: a MOLE Brain Board, a MOLE Personality Board, and software for a host computer. The host may be an IBM®-PC, or one of a number of inexpensive PC compatibles. The cross-assemblers and cross-compilers provided by National Semiconductor will run under control of the host computer MS-DOS operating system.

The Brain Board provides the MOLE system with the capability of communicating with the user's Host CPU. Resident firmware on the Brain Board allows the user to download assembled load modules over the RS-232 link from the host computer, display and alter code in both hex and mnemonic format, initiate Breakpoints, Traces, and timing on addresses and external events, examine and modify the internal resources of the Microcontroller being emulated. The Brain Board also provides all the hardware and firmware to program standard EPROMs up to 27256's (32k x 8).

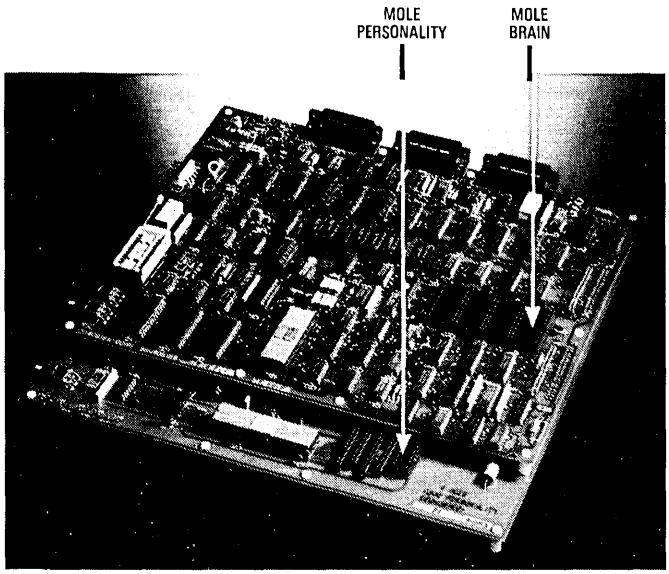
Development system flexibility is provided by the Personality board. This component tailors the system to emulate a single microcontroller family or device. For instance, one Personality Board supports the COP400 CMOS and NMOS family. This Personality Board provides emulation capability for 42 Microcontroller device types.

Personality boards are also available for the HPC and COPS family of M²CMOS products.

The host CPU contributes cost effective bulk storage and high speed processing. Disk editing and assembly operations are controlled by the host CPU. The results are down loaded to the Brain Board over the RS-232 link.

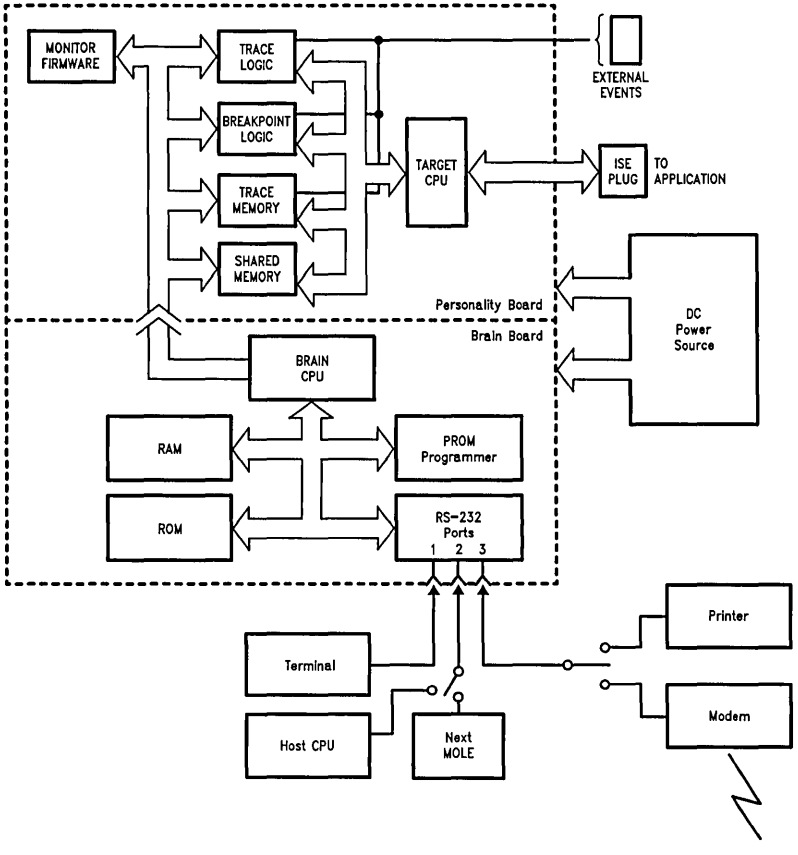
Once the application program has been completely debugged, the code may be submitted to National Semiconductor for use in creating the tooling necessary for manufacturing the masked Microcontroller device.

The MOLE concept provides the user with a powerful development system based around a familiar host. The Brain Board/Personality Board/Host combination provides FULL emulation capability. This modular design provides maximum flexibility and maximum utility for the development of Microcontroller based systems.



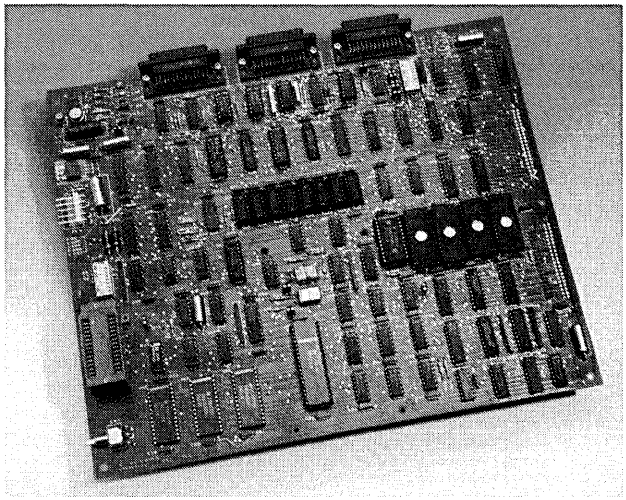
TL/DD/8830-3

MOLE System Block Diagram



TL/DD/8830-2

MOLE BRAIN BOARD



TL/DD/8830–19

GENERAL DESCRIPTION

The Brain Board is the pivotal component of the MOLE concept. In conjunction with a terminal and Personality Board it provides the user with a freestanding workstation for Microcontroller emulation. It ties the system together by communicating with the Personality Board, printers, modems, optional host computer, and other Brain Boards. Multiple Brain Boards, tied to a common host, can function as emulators for individual projects where each Brain Board is a separate workstation. They can also function as individual Microcontroller emulators within a multicontroller system.

The MOLE Brain Board utilizes a NSC800™ Microprocessor with 64k RAM and firmware ROM. It has an EPROM /EEPROM programmer for on-line changes. There are three RS-232 ports and a bus to connect the Brain to the Personality Board for actual emulation of code in the user's application system.

The RS-232 ports are used via the communication routines in firmware to interface with a host computer, terminal, modem, printer, or other MOLEs, for greater flexibility during system development.

The MOLE firmware is controlled by an EXEC. There are three major sets of EXEC commands. The first set of commands are calls to other main programs. These are:

COMM	Invoke Communications Program
DIAG	Invoke Diagnostics Program
MONITOR	Invoke Personality Emulation Monitor
PROG	Invoke PROM Programming Program

The second set of EXEC commands are:

CALC	Adds/Subtracts decimal and hex numbers
COMPARE	Compares one buffer with another
ERASE	Used to erase all or part of a buffer
HELP	Prints a summary of EXEX commands
MOVE	Moves data from one buffer to another
STATUS	Display status of buffers, display and alter RS-232 parameters

The third set of commands are used exclusively for multiple MOLE configurations and they are:

CONNECT	Connect the user with the requested system
DISCONNECT	Disconnects the MOLE
IDENT	Identifies a MOLE system

The MOLE Brain Board supports NSC's entire family of MOLE Personality boards.

FEATURES

- Single 5V operation
- Ability to interface to host computers
- Full communication control of other MOLEs with host computer and a modem
- Three RS-232 ports
- Auto baud selection (110, 300, 600, 1200, 2400, 4800, 9600, 19200 baud)
- Self diagnostics
- Program EPROMS
  - — MM2716, NMC27C16
  - — MM2732, NMC27C32
  - — NMC2764—NMC27256
- Program EEPROMs
  - — 9816
- Program emulator devices

PHYSICAL SIZE
10" x 12"

POWER REQUIREMENTS
+5V DC @ 3.5A
+12.5/+21V or +25V @ 50 mA
(Optional—required only for PROM programming)

ORDER P/N:
MOLE-BRAIN

MOLE-BRAIN PACKAGE CONTAINS
MOLE Brain Board
MOLE Brain User's Manual
2 RS-232 Cables
Power Cable
Miscellaneous Hardware

MOLE PERSONALITY BOARDS

The Personality Board lends personality to the MOLE system. The Monitor debugger firmware that is resident on the Personality Board is customized for the microcontroller that the Personality Board is designed to emulate, thereby giving the MOLE "personality". The Monitor firmware allows the user to display the application program in either hex or mnemonic format. The user can alter or deposit hex data into the program memory. A one-line assembler is also available to allow the user to put new instructions into the application program. Breakpoint, Singlestep, Trace or Time functions are available. They allow triggering on addresses or external events. The Monitor also provides the ability to examine and modify the internal RAM and registers of the Microcontroller being emulated.

Each Personality Board has its own Monitor; however, each Monitor implements a standard set of MOLE functions. This gives all MOLE systems a common set of functions with identical syntax. This commonality is designed to help provide a clear and simple migration path from the low-cost COP400 4-bit microcontrollers to the high performance HPC 16-bit microcontrollers without the need to relearn the development tool.

MOLE DEBUG FEATURES

The standard set of MOLE functions common to all MOLE Personality Boards is as follows.

TABLE I. Common MOLE Monitor Commands

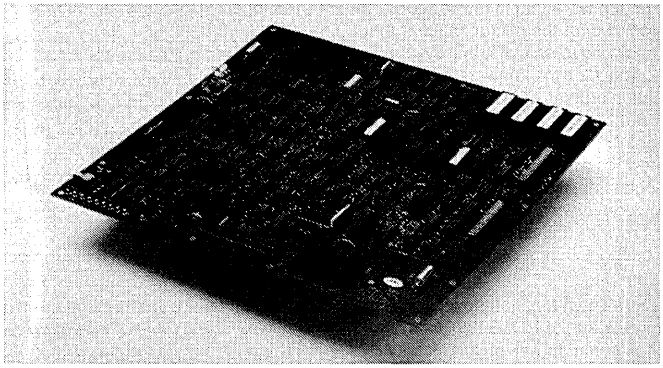
Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
Diagonstic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and MOLE Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace data
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory

These commands are implemented on the HPC, COP800 and COP400 MOLEs.

Additionally, each Personality board has its own special Monitor functions that give that system additional capabilities.

MOLE COP400 FAMILY PERSONALITY BOARD

COPS Personality Board



TL/DD/8830–6

GENERAL DESCRIPTION

The MOLE COPS Family Personality Board supports the emulation of COP400 family of Microcontrollers. The Personality Board allows the user to emulate the appropriate Microcontroller in the user's end system for fast development of application code and hardware. The Personality Board consists of: a Monitor, the hardware to control the operation of the Microcontroller in the emulation system, and an emulation cable to connect the emulator to the application system. The cable has the same pin configuration as the final masked part.

The Personality Board Monitor is contained in firmware ROM, contains an assembler and disassembler and is directly executable by the NSC800 on the Brain Board. The Monitor commands will allow the user to execute the application code, examine and modify internal registers and I/O, examine and alter object code in hex or mnemonic format, execute Time measurements, and set Trace and Breakpoints.

The Personality Board also contains 2k bytes of shared memory (RAM) for application code and the necessary hardware for Trace and Breakpoint operation.

FEATURES

- Supports entire COPS CMOS and NMOS family
- Single 5V operation
- Firmware monitor directly executed by Brain CPU
- Firmware diagnostics directly executed by Brain CPU
- Firmware Line Assembler and Unassembler
- 2k bytes of shared memory
- 256 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges, or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Common MOLE Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
Diagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and MOLE Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory

COP400 Monitor Special Functions

Chip	Specify COP device to emulate
Option	Specify COP chip options being emulated
Set	Set special emulation options

PHYSICAL SIZE
12″ x 12″

POWER REQUIREMENTS
+5V @ 3.5A

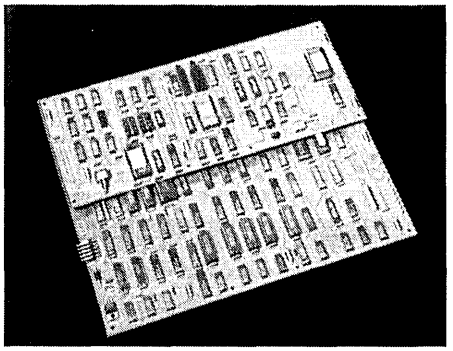
ORDER P/N:
MOLE-COPS-PB1

MOLE-COPS-PB1 PACKAGE CONTAINS
MOLE CMOS COPS Personality Board
MOLE CMOS COPS PB Manual
3 Emulator Cables
Power Cable
Miscellaneous Hardware

SOFTWARE ORDERED SEPARATELY
See How To Order

MOLE COP800 FAMILY PERSONALITY BOARD

COP800 Personality Board



TL/DD/8830–18

GENERAL DESCRIPTON

The COP800 Family Personality Board allows the MOLE system to emulate the COP800 family. The Personality Board consists of a firmware Monitor, 16k bytes of shared memory, 2000 deep Trace memory, Port recreation logic to recapture the pins used for emulation, emulation hardware, and an In System Emulator (ISE) cable. The ISE cable has the same pinout as a socketed masked part and allows the Personality Board to function within the application system.

The NSC800 CMOS Microprocessor, located on the Brain Board, directly executes the Personality Board Monitor firmware. The Monitor allows execution of application code, examination and alteration of internal registers, examination and alteration of shared memory, and the setting of trace and breakpoints. The ISE cable connects these capabilities to the application system. Up to eight external events as well as 15-bit address and 8-bit data busses can be traced in the 2000 deep trace memory. Multiple breakpoints, plus assemble and unassemble commands are at the user's disposal.

Application programs of up to 32k bytes from Personality Board RAM may be emulated.

FEATURES

- Supports COP800 microcontroller family
- Single 5V operation
- Firmware monitor directly executed by Brain CPU
- Firmware diagnostics directly executed by Brain CPU
- Firmware Line Assembler and Unassembler
- 32k bytes of shared program memory
- 2000 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Common MOLE Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
DIagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and MOLE Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Diassembler for Trace or shared memory

COP8 Monitor Special Functions

CYcles	Capture COP8 execution cycles Trace memory
End	Exit Monitor and return to Brain Exec
EXclusion	Specify address ranges to exclude from Trace
ListUnassemble	List shared memory in mnemonic form
TypeUnassemble	Type Trace memory in mnemonic form

PHYSICAL SIZE
12" x 12"

POWER REQUIREMENTS
+5V @ 3.5A

ORDER P/N:

MOLE-COP8-PB1	**COP820/840**
MOLE-COP8-PB2	**COP888**

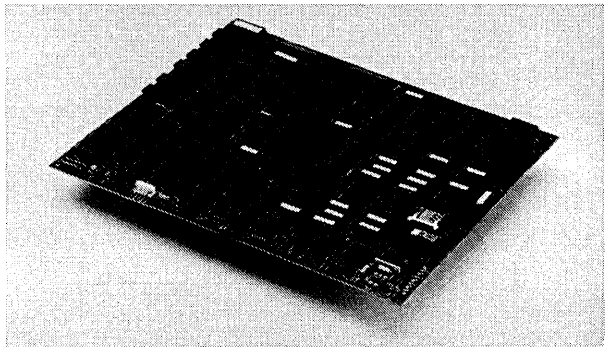
MOLE-COP8-PB1/2 PACKAGE CONTAINS
MOLE CMOS COP8 Personality Board
MOLE CMOS COP8 PB Manual
Emulator Cables
Power Cable
Miscellaneous Hardware

SOFTWARE ORDERED SEPARATELY
See How To Order

8

MOLE HPC FAMILY PERSONALITY BOARD

HPC Personality Board



TL/DD/8830–10

GENERAL DESCRIPTION

The HPC Family Personality Board allows the MOLE system to emulate the High Performance Controller (HPC) family. The Personality Board consists of a firmware Monitor, 16k bytes of shared memory, 2k x 48 Trace memory, Port recreation logic to recapture the pins used for emulation, emulation hardware, and an In System Emulator, ISE, cable. The ISE cable has the same pinout as a socketed masked part and allows the Personality Board to function within the application system.

The NSC800 CMOS Microprocessor, located on the Brain Board, directly executes the of Personality Board Monitor firmware. The Monitor allows execution of application code, examination and alteration of internal registers, examination and alteration of shared memory, and the setting of trace and breakpoints in either shared or user memory. The ISE cable connects these capabilities to the application system. Up to eight external events as well as 16-bit address and 16-bit data busses can be traced in the 2k deep trace memory. Multiple breakpoints, and chip error conditions plus assemble and unassemble commands are at the user's disposal.

Applications programs of up to 16k bytes from Personality Board RAM or 64k bytes from user system RAM may be emulated.

FEATURES

- Supports HPC microcontroller family
- Single 5V operation
- Firmware monitor directly executed by Brain CPU
- Firmware diagnostics directly executed by Brain CPU
- Firmware Line Assembler and Unassembler
- 16k bytes of shared program memory
- 2000 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- List and alter display memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Common MOLE Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
DIagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and MOLE Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory

HPC Monitor Special Functions

AlterWord	Alter consecutive words in shared memory
BAnk	Specify bank trigger information
CHip	Select chip and specify system memory map
DepositWord	Deposit word value in range of shared memory
End	Exit Monitor and return to Brain Exec
ERror	Enable/disable HPC access error checking
EXclusion	Specify address ranges to exclude from Trace
FindWord	Find word values in shared memory
ListWord	List shared memory or memory range as words
MAp	Specify address range of memory on-board MOLE
XMove	Move data from one address range to another

PHYSICAL SIZE
12" x 12"

POWER REQUIREMENTS
+5V @ 8A

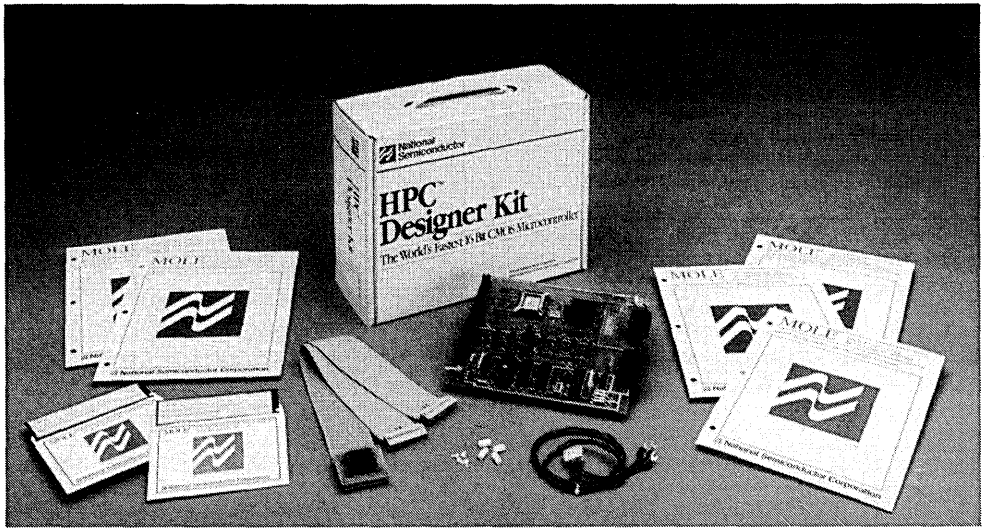
ORDER P/N:
MOLE-HPC-PB1

MOLE-HPC-PB1 PACKAGE CONTAINS
MOLE HPC Personality Board
MOLE HPC PB User's Manual
1 Emulator Cable
Power Cable
Miscellaneous Hardware

SOFTWARE ORDERED SEPARATELY
See How To Order

HPC DESIGNER'S KITS

HPC Designers Kits



TL/DD/8830-20

GENERAL DESCRIPTION

The HPC Designer Kits are a 16-bit microcontroller Development System for program development and real-time emulation. An on-board HPC microcontroller executes monitor firmware and also acts as the target processor.

When used as the target processor, all of the features of the HPC are available for use in the application. All operating modes of the HPC are supported, with up to 64k bytes of addressable memory available for application programs.

This kit contains all of the components, manuals, and software to design an HPC system. Just add an IBM or compatible PC, +5V DC 1.5-Amp power supply and RS232 cables.

Two kits are offered. The evaluation package contains evaluation software that allows up to 1000 lines of code to be assembled and linked. The development package has a complete Assembler/Linker/Librarian with no code limitations.

FEATURES

- Supports HPC microcontroller family
- Single 5V operation
- Firmware monitor directly executed by the HPC
- Firmware diagnostics directly executed by the HPC
- Firmware Line Assembler and Unassembler
- 64k bytes of addressable program memory
- Breakpoint on multiple addresses
- List and alter memory
- Print and modify internal registers
- Singlestep
- Real time emulation
- Evaluation module that allows up to 1000 lines of code to be developed for evaluation purposes

HPC Development Board Monitor

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
BAud	Set or display the host or terminal Baud rate
BYpass	Connect terminal port to host port
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint function
Deposit	Deposit byte value into range of shared memory
DIagonstic	On-board test routine for system checkout
Go	Start program execution
Help	On-screen Help menu
List	List data in shared memory
ListUnassemble	List shared memory in mnemonic form
LOad	Load hex object file from terminal or host port
Modify	Modify on-chip RAM or Registers during Breakpt
ModifyByte	Modify on-chip RAM or registers as bytes
ModifyWord	Modify on-chip RAM or registers as words
Put	One-line assembler
Restart	Restart HPC chip, same as Reset on the MOLE
Singlestep	Execute one instruction, then Breakpoint
Type	Type on-chip data during Breakpoint
Unassemble	Disassembler for shared memory

PHYSICAL SIZE
12" x 12"

POWER REQUIREMENTS
+5V @ 1.5A

ORDER P/N:
HPC-MOLE-EVAL0 (Evaluation Package)
HPC-MOLE-DEVL0 (Development Package)

MOLE-HPC-EVAL0 PACKAGE CONTAINS
HPC Evaluation Board
ISE Cable w/connector for PGA socket
Development Board Communications Software (MS-DOS)
HPC Assembler/Linker/Evaluation Software
C Compiler Evaluation Module Software
HPC46083/46043/46003 User's Manual
HPC46083/46043/46003 Datasheet
Dial-A-Helper User's Manual

MOLE-HPC-DEVL0 PACKAGE CONTAINS
HPC Evaluation Board
ISE Cable w/connector for PGA socket
Development Board Communications Software (MS-DOS)
HPC FULL Assembler/Linker/Librarian Software
C Compiler Evaluation Module Software
HPC46083/46043/46003 User's Manual
HPC46083/46043/46003 Datasheet
Dial-A-Helper User's Manual

MOLE SYSTEMS

HOW TO ORDER MOLE SYSTEMS

MOLE systems are available for a variety of microcontrollers. To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Included, along with the cross assembler, in the software package are two file conversion routines to convert the assembler output (LM) to HEX and to convert HEX to LM. Also included in the software package is a COMM program which facilitates the downloading and uploading between the host and the MOLE, and adds the capability to make the host act as a terminal.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBMR	Relocatable Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001
	424410897-001	Users Manual		424410897-001
COP820/840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Users Manual		420410703-001
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
COP400	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COPS-PB1	Personality Board	COP400 Personality Board Users Manual	420408189-001
	MOLE-COPS-IBM	Assembler Software for IBM	COP400 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424409479-002 420040416-001
	424410284-001	Users Manual		424410284-001

DESIGNER KITS

HOW TO ORDER DESIGNER KITS

Designer Kits are self contained development systems that contain all of the components, manuals and software to design a microcontroller based system. Just add an IBM-PC or compatible PC, +5V DC 1.5 Amps power supply and RS232 cables.

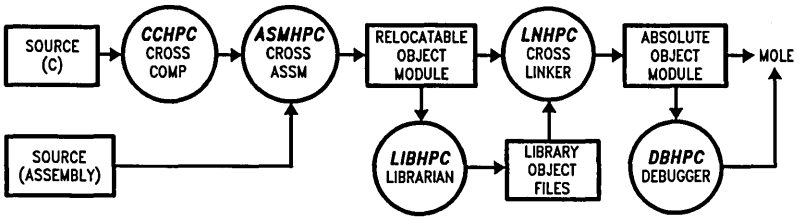
Two types of kits are offered. The Evaluation package contains evaluation software that allows limited code to be developed. The Development package has no restrictions on the assembler software.

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-HPC-EVAL0	HPC Designer's Kit Evaluation Version	HPC-DB1 Board Evaluation Compiler, Assembler/Linker, Manuals	420410901-1
	MOLE-HPC-DEVL0	HPC Designer's Kit Development Version	HPC-DB1 Board Evaluation Compiler, FULL Assembler/Linker, Manuals	420410901-1

DEVELOPMENT SYSTEM ACCESSORIES AND REPLACEMENT PARTS

Part Type	Order Part Number	Description
MOLE EMULATOR CABLES		
68-Pin PGA Cable	MOLE-CBL-68PGA	Cable used for in-system emulation of the HPC in a 68 PGA package. For the HPC MOLE.
68-Pin PLCC Cable	MOLE-CBL-68PCC	Cable used for in-system emulation of the HPC in a 68 PLCC package. For the HPC MOLE.
44-Pin PLCC Cable	MOLE-CBL-44PCC	Cable used for in-system emulation of the COP8 in a 44 PLCC package. For the COP888 MOLE.
28-Pin PLCC Cable	MOLE-CBL-28PCC	Cable used for in-system emulation of the COP8 in a 28 PLCC package. For the COP8 MOLE.
40-Pin DIP Cable	MOLE-CBL-40DIP	Cable used for in-system emulation of the COP8 in 40-pin DIP packages. For the COP8 MOLE.
28-Pin DIP Cable	MOLE-CBL-28DIP	Cable used for in-system emulation of COP4 and COP8 devices in the 28-pin DIP package. For use with the COP4 and COP8 MOLEs.
24-Pin DIP Cable	MOLE-CBL-24DIP	Cable used for in-system emulation of COP4 and COP8 devices in the 24-pin DIP package. For use with the COP4 and COP8 MOLEs.
20-Pin DIP Cable	MOLE-CBL-20DIP	Cable used for in-system emulation of COP4 and COP8 devices in the 20-pin DIP package. For use with the COP4 and COP8 MOLEs.
SUPPORT PRODUCTS		
COP444 PIG	COP444CP	A piggy-back emulator product designed to provide programmable form, fit and function emulation for the COP4XXC products in a 28-lead DIP package. An 8k x 8 EPROM sits piggy-back in a socket on top of a hybrid packaged 28-lead COP404C controller.
COP820/840 PIG	COP820CP-X COP840CP-X	A piggy-back emulator product designed to provide programmable form, fit and function emulation for the COP820 and COP840 products in a 28-lead DIP package. An 8k x 8 EPROM sits piggy-back in a socket on top of a hybrid packaged 28-lead COP820/840 controller. X is the clock option (from datasheet).
COP8720 Programmer	MOLE-COP8-PROG	Adapter board for use in programming the COP8720, 8721 or 8722 devices on the MOLE-Brain board.
COP888 PIG	TBD	A piggy-back emulator product designed to provide programmable form, fit and function emulation for the COP888 family.
HPC PCB Emulator	HPC16083MH	A form, fit and function programmable emulator for the 68-lead PLCC HPC16083 device used in single-chip mode. Programmed with an adapter board on the MOLE-Brain.
HPC16083MH Programmer	MOLE-HPC-PROG	Adapter board for programming the HPC16083MH.
SYSTEM HARDWARE		
MOLE-Brain	MOLE-BRAIN	Main board component of the MOLE Development System.
MOLE Enclosure	TBD	Kit for complete enclosure of the HPC and COP8 MOLE systems. Includes box, power supply and all cabling required to upgrade existing MOLE Systems.
MOLE SOFTWARE SUPPORT FOR THE IBM-PC		
HPC Assembler	MOLE-HPC-IBMR	Relocating ASMHPC Asembler/Linker/Librarian.
HPC C Compiler	MOLE-HPC-IBM-CR	CCHPC C Compiler. Includes the HPC Assembler.
HPC Evaluation Software	MOLE-HPC-IBMEVL	HPC Evaluation software. Includes: ASMHPC and CCHPC evaluation modules and manuals.
COP8 Assembler	MOLE-COP8-IBM	COP800 Assembler.
COP4 Assembler	MOLE-COPS-IBM	COP400 Assembler.
Dial-A-Helper	MOLE-DIAL-A-HLP	Dial-A-Helper manual and communications software.

HPC™ Software Support Package



TL/DD/9727–1

- **Choice of host systems**
  - **IBM® XT/AT PC-DOS**
  - **VAX™ VMS™**
  - **VAX UNIX®**
- **CCHPC C Compiler**
  - **ANSI Draft Standard C (February 1986)**
  - **Additional storage class modifiers supported**
  - **Additional statement types included**
  - **Supports embedded assembly code**
  - **Supports multiple source files**
- **ASM HPC Assembler**
  - **Macro and conditional assembly**
  - **Instruction size optimization**
  - **Symbol table and cross reference output**
  - **Object files are linkable and relocatable**
- **LNHPC Linker**
  - **Links multiple relocatable object modules**
  - **Selects required modules from library files**
- **LIBHPC Librarian**
  - **Supports user developed library modules**
- **DBHPC Source Debugger**
  - **High level software debugging**
  - **Real-Time Hardware Emulation**
  - **Source file listing**
  - **Variable set and view**
  - **Set break points at C source level**
  - **Display dynamic function nesting**
  - **Display debugger status**
  - **Call operating system commands**

General Description

The HPC software support packages provide development system support for the HPC family of 16-bit single chip microcontrollers. Two software packages are offered that support the HPC: HPC Assembler/Linker/Librarian and HPC C Compiler. Both packages are available for a choice of host systems: IBM XT/AT PC DOS, VAX VMS and VAX UNIX.

The assembler produces relocatable object modules from the HPC macro assembly language instructions. The object modules are then linked and located to absolute memory locations. The absolute object module may be downloaded to the HPC MOLE™ (Microcontroller OnLine Emulator) development system for debugging.

The C compiler generates assembly source. The C Compiler may optionally pass symbolic information through the assembler and linker to the absolute object module. The source debugger then uses this information for C and Assembly language debugging on the host in conjunction with the MOLE.

HPC C Compiler—CCHPC Introduction

The HPC C Compiler (CCHPC) is a full and complete implementation of ANSI Draft Standard C (Feb 1986) for freestanding environment. Certain additions are included to take advantage of special features of the HPC (for the specific needs of microcontrollers). The Enhancements include the support of two non-standard statement types (loop and switchf), non-standard storage class modifiers and the ability to include assembly code in-line. The compiler supports enumerated types of structures by value, functions returning structures, function prototyping and argument checking.

Symbol Names, both internal and external, are 32 characters. Numerics are 16-bit for **short** or **int**, 32-bit for **long**, and 8-bit for **char**, all as either **signed** or **unsigned**; floating point is offered as **float** of **double**, both using IEEE format.

All data types, storage classes and modifiers are supported. Additional storage class modifiers are provided:

BASEPAGE place **static** variable in faster and more efficient on-chip basepage memory.

NOLOCAL declare function without local variables, thus no stack frame.

INTERRUPTn declare function to execute in response to specific interrupt(s).

ACTIVE declare function to be accessed via faster and more efficient function call mechanism.

All statement types are supported, and two additions are provided:

loop (count) simpler, more efficient **for** looping command.

switchf (value) faster form of **switch** command without constraint checking.

CCHPC SPECIFICATIONS

Note: Enhancements are boldface.

Name length — 32 letters, 2 cases

Numbers

- Integer, Signed and Unsigned — 16–32 bits
- Short and Long — 16 bits and 32 bits
- Floating, Single and Double — 32 bits and 32 bits

Preprocessor

#include
#define #define() #undef
#if #ifdef #ifndef #if defined #else #elif #endif

Declarations

auto register const volatile **BASEPAGE**
static static global static function **NOLOCAL INTERRUPTn ACTIVE**
extern extern global extern function
char short int long signed unsigned float double void
struct union bit field enum
pointer to array of function returning
type cast typedef initialization

Statements

; { . . . } expression ; assignment ; structure assignments ;
while () . . . ; do . . . while () ; for(; ; ;) . . . ; **loop () . . . ;**
if () . . . else . . . ; switch () . . . ; case : . . . ; default : . . . ; **switchf () . . . ;**
return ; break ; continue ; goto . . . ; . . . :

Operators

primary: function() array[] struct_union . struct_pointer ->
unary: * & + - ! ~ ++ -- sizeof (typecast)
arithmetic: * / % + - << >>
relational: < > <= >= == !=
boolean: & ^ | && ||
assignment: = += -= *= /= %= >>= <<= &= ^= |=
misc.: ?: ,

Functions

- arguments: Numbers, Pointers, Structures
- return values: Numbers, Pointers, Structures

forward reference (argument checking)

Library Definition **Limited-Freestanding environment**

Embedded Assembly Code

HPC C Compiler—CCHPC Introduction (Continued)

All operators are supported, and anachronisms have been eliminated (as per the standard). Structure assignment, structure arguments, and structure functions are also supported. Forward reference functions and argument type checking is supported.

Assembly code may be embedded within C programs between special delimiters.

COMPILER COMMAND FEATURES

The CCHPC runs under different host operating systems. Depending on the host system and the CCHPC command line options, ordering of the elements and their syntax may vary. In all cases, the command line consists of the command name, options or switches, and the filename to be compiled.

The compiler output, in the form of ASMHPC assembler source statements, is put in a file with the extension ".asm".

The following is a description of the CCHPC options or switches:

Include C code in assembler code output—Assembler output file contains the C source code lines as comments.

Invoke C preprocessor before compilation—Allows the C preprocessor invocation to be skipped.

Invoke an alternative C preprocessor before compilation—Allows an alternative preprocessor to be used.

Setting the stack size—This switch takes a numeric argument in the form of a C constant. If the module being compiled contains the function main, the compiler uses the number as the size of the program's execution stack, in words. The option is ignored if the module does not contain main.

Creating 8-bit wide code—This switch creates code that can be executed from 8-bit wide memory by avoiding the use of instructions that fetch 16-bit operands (such as JIDW). This option DOES NOT allow the use of 16-bit values or data in 8-bit memory.

Placing string literals in ROM—The ANSI draft language standard calls for string literals, and individual copies for each usage of the literal to be stored in RAM. This switch allows CCHPC to override this requirement for efficiency, saving startup time, RAM and ROM space. Turn off compiler warning messages.

Indicating directories for include files—This switch takes a string argument which is passed to the C preprocessor. The C preprocessor uses it as a directory to search for include files.

Defining symbol names—This switch passes the string argument to the C preprocessor. It instructs the preprocessor to perform the same function as the #define, allowing the symbol definitions to be moved to the invocation line.

Undefining symbol names—Similarly, this switch passes a string argument to the C preprocessor. It removes any previous definitions.

Permit old-fashioned constructs—Certain anachronisms from Kernighan and Ritchie C that are not permitted in ANSI C will be accepted by the compiler if this option is specified. This option is a convenience for users porting a C program to CCHPC from a Kernighan and Ritchie compiler.

Set chip revision level—This switch is used to generate code to work around bugs in specified chip revisions.

Generate symbolic debug information—This option causes the compiler to create symbolic debug information which is passed to the output assembly file.

BASIC DEFINITIONS

Names may be arbitrarily long, but only the first 32 characters are significant. Case distinctions are respected.

Constants may be of type decimal, octal, hex, character and string.

Escape sequences for new line, horizontal and vertical tab, backspace, carriage return, form feed, alert, backslash, single quote, double quote, octal and hexidecimal numbers are supported.

Comments imbedded in the source code begin with "/*" and end with "*/". Comments can not be nested.

CCHPC supports the following Data types:

Name	Size in Bits
char	cc8
short	16
int	16
enum	8 or 16
long	32
signed char	8
signed short	16
signed int	16
signed long	32
unsigned char	8
unsigned short	16
unsigned int	16
unsigned long	32
float	32
double	32
long double	32
struct	sum of component sizes
union	maximum of component sizes

The type "char" is treated as signed. Unsigned operations are treated the same as signed operation, except for multiplication, division, remainder, right shifts and comparisons. For signed integers, the compiler uses an arithmetic right shift. For unsigned integers, a logical shift is used when shifting right.

HPC C Compiler—CCHPC Introduction (Continued)

Keywords const and volatile can be applied to any data. Const indicates that the symbol refers to a location which is read-only. If the symbol is in static or global storage, it will be assigned to ROM memory. Volatile indicates that optimization must not change or reduce the accesses to the symbol.

Since the HPC supports 8-bit operations, CCHPC does not automatically promote "char" types to "int" when evaluating expressions. For a binary operation, the compiler promotes a "char" to an "int" only if the other operand is a 16-bit (or more) value or if the result of the operation is required to be a 16-bit (or more) value. The use of 8-bit operations yields efficient code without compromising the correctness of the result.

CCHPC uses the standard C preprocessor and any standard preprocessor functions, including "#define", "#include" and macros with arguments are supported.

A program is set of intermixed variable and function definitions. Variables must always be defined before use, functions may be defined in any order.

Variable initialization is performed according to the draft ANSI standard rules.

Standard C operators, and their hierarchy are as described in the ANSI standard draft.

CCHPC allows the programmer to imbed assembler code directly in the C source. All data between "/$" and "$/" is copied directly to the assembler output file generated by CCHPC.

CCHPC IMPLEMENTATION DEPENDENT CONSIDERATIONS

Memory

CCHPC is designed to execute in a 16-bit environment. Special care must be taken when using CCHPC in an 8-bit HPC system.

Storage Classes

CCHPC supports the following storage classes:

auto
static
register
typedef
extern

Due to HPC architectural features, the "register" storage class is limited. A variable can be assigned a "register" only if it is of type pointer and only if a register is available. The first "register" pointer variable encountered is assigned to the HPC B register, the second to the HPC X register and any subsequent ones are treated as "auto" (unless NOLOCAL is in effect, in which case it will be treated as "static").

The default storage class for global declarations is "static". The default storage class for declarations within functions is "auto".

Storage Class Modifiers

To make maximum efficient use of HPC architectural features CCHPC supports the notion of "storage class modifiers". A storage class modifier may appear with or in place of a storage class. Following is the set of storage class modifiers:

Keyword	Applicable to
BASEPAGE	variable
ACTIVE	function
NOLOCAL	function
INTERRUPTn (where n = 1 to 7)	function

Storage class modifiers may be supplied with each variable or function declaration. The effect of each storage class modifier is described in the following:

BASEPAGE—The variable will be allocated in the BASE section. Accessing a basepage variable is more efficient than accessing any other type of variable but the amount of basepage storage is limited.

ACTIVE—The address of the function is placed in the 16 word JSRP table. Calls to the function will require 1 byte of code. The most frequently called functions should be considered for designation as ACTIVE functions for maximum code efficiency.

NOLOCAL—The functions local variables are not allocated on the run-time stack. Instead, they are allocated in static storage. Access to local variables in a NOLOCAL function will be more efficient since access can be direct rather than indexed from the frame pointer. If a function has no arguments or local variables, then entry and exit from the function will be much more efficient since there will be no need to adjust the frame pointer on entry and exit of the function.

INTERRUPTn—These modifiers can be used to set interrupt vectors (one through seven) to point to a particular function. Any function which has an INTERRUPT storage class modifier has special entry and exit code generated. This code will push all HPC registers (A, B, K, X, PSW and word at RAM address 0) onto the stack before executing normal function entry code. Exit code restores all registers before returning from the interrupt.

C Stack Formation

The Stack Pointer (SP) is initialized to the start address assigned by the linker. The Stack Pointer always points to the next free location at the top of the stack.

Within a function, the compiler maintains a Frame Pointer which is used to access function arguments and local automatic variables. The Frame Pointer location is reserved by the compiler at location Oxbe.

HPC C Compiler—CCHPC Introduction (Continued)

To call a function, the compiler pushes arguments onto the stack in reverse order, performs a jump subroutine to the function, then decrements the Stack Pointer by the number of bytes pushed onto the stack. Since all stack pushes are 16-bits, any 8-bit arguments are automatically promoted to 16-bits. On function entry, the compiler creates new stack and frame pointers for the function. On exit, the stack and frame pointers are restored to the values they had on entry to the function.

Using In-Line Assembler Code

CCHPC allows in-line assembler code to be entered in the body of a C function. The assembler code can access any of the currently active variables or can get the address of a variable.

Efficiency Considerations

HPC code size and execution time can be optimized by making maximum use of BASEPAGE variables. When BASEPAGE is full, static variables are next most efficient. The least efficient variables are automatic since they require an indirect indexed access. Minimizing the use of longs and floats will improve efficiency. The HPC architecture strongly supports unsigned arithmetic, so the programmer should use unsigned variables except for cases that absolutely require signed arithmetic. The compiler does not attempt to identify common subexpressions for computation only once, so this must be done by the programmer.

Statements and Implementation

The following C statements are supported by CCHPC:

expression;
if
if . . . else
while . . .
do . . . while
for . . .
break
goto
continue
return
return . . .
case . . .
default
switch . . .
switchf . . .
loop . . .

The switch statement will generate an efficient jump table for a set of cases if the cases are sufficiently close, or it will generate individual tests for each case. The switchf statement is the same as the switch statement except that when a jump table is generated for the switchf statement the compiler does not generate the code necessary to check the bounds of the value to be switched on. This creates a more efficient form of the switch statement but the programmer must insure that the value being switched on is in range.

The loop statement is an extension to the ANSI standard. Loop allows the programmer to create a code efficient loop by using the HPC DECSZ instruction. The loop statement may be nested. A break statement inside the loop will cause an immediate exit from the loop.

Run-Time Notes

During evaluation of complex expressions, the compiler uses the stack to store intermediate results.

All HPC C programs start with a call to the function "main" with no arguments. Before calling "main", run-time start-up code initializes RAM. The initial values of static or global variables with initialization are stored in ROM and copied to the appropriate variables in RAM. Static or global variables without initialization are cleared to zero. The function "main" must be defined. When "main" returns to the run-time start-up routine it executes the HALT macro provided which puts the chip in an infinite loop.

Since the run-time stack is of fixed size and there is no check for stack overflow, it is up to the programmer to insure that the stack area is large enough to prevent stack overflow.

Memory location zero is reserved by the compiler.

The HPC C Compiler User's Manual provides additional information on the features and functions of CCHPC.

HPC Cross Assembler—ASMHPC

INTRODUCTION

The MOLE HPC cross-assembler (ASMHPC) is a cross-assembler for the NSC HPC family of microcontrollers. ASMHPC translates symbolic input files into object modules and generates an output listing of the source statements, machine code, memory locations, error messages, and other information useful in debugging and verifying programs.

ASMHPC has the following useful features—

— Macro capability that allows common code sequences to be coded once.

— Conditional code assembly is supported.

— Translates symbolic assembly code modules into object code. Object modules are linkable and relocatable.

— Symbolic names may be defined for any HPC register, memory location or I/O port. Symbols may be defined as byte or word size.

— Symbol table and cross-reference output is provided.

— Full set of Assembler directives are provided for ease of generating vector tables for interrupts, short subroutine calls, jump indirects and other data generation within the object program.

— Data and code sections are user definable. Sections may be relocatable or absolute. Sections

HPC Cross Assembler—ASMHPC (Continued)

may be assigned to 8-bit memory to support the HPC 8-bit mode. Data sections may be assigned to basepage RAM on the HPC to maximize efficient access to variables.

— Accepts assembly source code generated by the HPC C Compiler, CCHPC.

— Full set of Assembler controls for greater flexibility in debugging modules and programs created by ASMHPC.

ASSEMBLY LANGUAGE ELEMENTS

Assembly Language Statement

Assembly language statements are comprised of four fields of information.

Label field—This is an optional field. It may contain a symbol used to identify a statement referenced by other statements. A symbol used in this manner is called a label.

Operation field—This field contains an identifier which indicates what type of statement is on the line. The identifier may be an instruction mnemonic or an assembler directive. The operation field is required on all assembler statement lines, except those lines which consist of only a label and/or comment.

Operand field—The operand field contains entries that identify data to be acted upon by the operation defined in the operation field. Operand examples are source or target addresses for data movement, immediate data for register initialization, etc.

Comment field—Comments are optional descriptive notes that are included in the program and listings for programmer reference and program documentation. Comments have no effect on the asembled object module file.

Character Set

Each assembly language statement is written using the following characters:

Letters—A through Z (a through z)
Numbers—0 through 9
Special Characters—!$%'( )*+,−./;:<=>&#?_b^

Note: Upper and lower case are distinct; b^ indicates a blank.

Location Counter

There is a separate location counter for each program section, and the counter is relative to the start of that section. The assembler uses the location counter in determining where the current statement goes in the current program section. If the program section is relocatable, the linker does the final job of assigning an absolute address to the instruction.

Symbols and Labels

Symbols and labels are used to provide a convenient name for values and statements. Symbols and labels have the same rules for construction, only their use distinguishes a symbol from a label.

Rules for symbol or label construction are:

1. The first character must be either a letter, a question mark (?), an underscore (_), a dollar sign ($) or a period (.).
2. All other characters may be any alphanumeric character, dollar sign ($), question mark (?) or underscore (_).
3. The maximum number of characters in a symbol or label may be selected by the user with the SIZE-SYMBOL control. The default is 64.
4. Symbols starting with dollar sign ($) are local symbols and are defined only within a local region.
5. Labels and symbols are case sensitive.

Operand Expression Evaluation

The expression evaluator in the assembler evaluates an expression in the operand field of a source program. The expressions are composed of combinations of terms and operators. An expression may consist of a single term or may consist of two or more terms combined using operators. Terms are—numbers in decimal, hexadecimal, octal or binary, string constants, labels and symbols or the location counter symbol. Each term has four attributes: its' value, relocation type, memory type and size. The relocation type is either absolute or relocatable. The memory type indicates whether the term represents a BASE, RAM8, ROM8, RAM16, ROM16 or null (in the case of an absolute term). The size of a term is null, byte or word.

The operators allowed in ASMHPC are: arithmetic, logical, relational, upper and lower byte extraction and untype operators. Arithmetic operators are +, −, *, /, MOD, SHL, ROL and ROR. The logical operators are NOT, AND, OR and XOR. The relational operators are EQ, NE, GT, LT, GE and LE. Upper and lower extraction operators are HIGH and LOW. The untype operator is &.

Parentheses are permitted in expressions. Parentheses in expressions override the normal order of evaluation, with the expression(s) within parentheses being evaluated before the outer expressions.

Numbers are represented in ASMHPC in 16-bit 2's complement notation. Signed numbers in this representation have a range of −32768 (x'8000) to +32767 (x'7FFF). Unsigned numbers are in the range of 0 to 65535. String constants are internally represented in the 8-bit ASCII code. All expression evaluation is done treating terms as unsigned numbers, for example, −1 is treated as having the value x'FFFF. The magnitude of the expression must be compatible with the memory storage available for the expression. For example, if the expression is to be stored in an 8-bit memory location, then the value of the evaluated expression must not exceed x'FF.

HPC Cross Assembler—ASMHPC (Cor tinued)

ASSEMBLY PROCESS

The ASMHPC assembler performs its functions by reading the assembly language statements sequentially from the beginning of a module or a program to the end, generating the object code and a program as it proceeds.

The ASMHPC assembler is a multi-pass assembler which allows it to resolve forward referenced symbols and labels efficiently. The number of passes can be selected using the PASS control. This allows the user to select the level of optimization of forward referenced instructions.

MACROS

Macros help make an assembly language program easier to create, read and maintain. A macro definition is an assembly statement or statements that are referred to by a macro name. The macro may have parameters that are operated upon by the assembly statements. ASMHPC will substitute the macro definition for the macro name with the appropriate parameters during the assembly process. Repetitive or similar code can be defined as macros and the programmer can use the macros to build a library of basic routines. Variables unique to particular applications can be defined in and passed to a particular macro when called by main programs.

Defining a Macro

Macros must be defined before they are used in a program. Macro definitions do not generate code. Code is generated only when the macros are called by the assembly program. Macro definitions have a Macro name by which the macro will be referred in the program, declaration of any parameters to be used in the macro, assembler statements that are contained in the macro and directives that define the boundaries of the macro.

Following is the macro definition structure:

```
.MACRO mname [,parameters]
      •
      •
      •
macro body
      •
      •
      •
.ENDM
```

where:

— .MACRO is the assembler directive which initiates the macro definition.

— mname is the name of the macro. Multiple macros can have the same name. The last macro defined is the macro definition used. Macro definitions are retained in the macro definition table; if the current macro is deleted by the .MDEL directive, the previous definition becomes active. If mname is the same as a valid instruction mnemonic, the macro name is used in place of the normal instruction.

— Parameters are the optional list of parameters used in the macro. Parameters are delimited from mname and additional parameters with commas.

— The macro body is a sequence of assembly language statements and may consist of simple text, text with parameters, and/or macro-time operators.

— .ENDM identifies the end of the macro and must be used to terminate the macro definition.

Calling a Macro

Once a macro has been defined, it may be called by a program to generate code. A macro is called by placing the macro name in the operation field of the assembly language statement, followed by the actual value of the parameters to be used (if any). The form of a macro call is:

mname [parameters]

where:

— mname is the previously assigned name in the macro definition and

— parameters are the optional list of input parameters. When a macro is defined without parameters, the parameter list is omitted from the call.

The macro call as well as the expanded macro assembly code will appear on the assembler listing if the appropriate controls are enabled.

Using Parameters

The power of a macro can be increased with the use of optional parameters. The parameters allow variable values to be declared when the macro is called.

When parameters are included in a macro call, the following rules apply to the parameter list:

1. One comma and zero or more blanks delimit parameters.
2. A semicolon terminates the parameter list and starts the comment field.
3. Single quotes (') may be included as part of a parameter except as the first character of a parameter.
4. A parameter may be enclosed in single quotes ('), in which case the quotes are removed and the string is used as the parameter. This function allows blanks, commas, or semicolon to be included in the parameter. To include a quote in a quoted parameter, include two quotes ('').
5. Missing or null parameters are treated as strings of length zero.

The macro operator @ references the parameter list in macro call. Using the operator @ in an expression, the number of parameters can be used to control conditional macro expansion. The @ operator may also be

HPC Cross Assembler—ASMHPC (Continued)

used with a constant or symbol to reference the individual parameters in the macro parameter list. These capabilities eliminate the need for naming each parameter in the macro definition, which is useful when there are long parameter lists. Using the @ parameter count operator it is possible to create macros which have a variable number of parameters.

The macro operator for concatenation is ^. In a macro expansion the ^ operator is removed and the strings on each side of the operator concatenated after parameter substitution. This operator provides the ability of creating variable labels through the use of macros.

Local Symbols

When a label is defined within a macro, a duplicate definition results with the second and each subsequent call of the macro. This problem can be avoided by using the .MLOC directive to declare labels local to the macro definition.

Conditional Expansion

The conditional assembly directives allow the user to generate different lines of code from the same macro simply by varying the parameter values used in the macro calls.

Nested Macro Calls

Nested macro calls are supported. A macro definition may call another macro. The number of allowable levels of nesting depends on the sizes of the parameter lists, but at least ten is typical.

A logical extension of the nested macro call is the recursive macro call, that is a macro that calls itself. This is allowed, but the programmer must insure that the call does not create an infinite loop.

Nested Macro Definitions

A macro definition may be nested within another macro. Such a macro is not defined until the outer macro is expanded and the nested macro is executed. This allows the creation of special purpose macros based on the outer macro parameters. Using the .MDEL directive and the nested macro capability a macro can be defined only within the range of the macro that uses it.

Macro Comments

All lines within a macro definition are stored with the macro, however, any text following ";;" is removed before being stored. This text will appear on the listing of the macro definition but will not appear on the macro expansion.

ASSEMBLY LISTING

The listing generated by ASMHPC contains program assembly language statements, line numbers, page numbers, error messages and a list of the symbols used in the program. The listing of assembly language statements which generate machine code includes the hexadecimal address of memory locations used for the statement and the contents of these locations. To the left of the instruction, an "R" indicates a relocatable argument in this instruction, "X" indicates an external argument, "C" indicates a complex argument and "+" indicates macro expansion.

The assembler listing optionally includes an alphabetical listing of all symbols used in the program together with their values, absolute or relocatable type, word or byte or null type, section memory type and public or external. Optionally a cross reference of all symbol usage by source line number is given; the defining line number is preceded by a "-".

The total number of errors and warnings, if any, is printed with the listing. Errors and warnings associated with assembly language statements are flagged with descriptive messages on the appropriate statement lines.

Directives

Directive statements control the assembly process and may generate data in the object program. The directive name may be preceded by one or more labels, and may be followed by a comment. The directive's name occupies the operation field. Some directives require an operand field expression.

Assembler Controls

An assembler control is a command that may be used in the source program on a control line or on the invocation line as an option. A control line is indicated by a # in column 1 of the source line. Comments may be included on a control line by preceding the comment with a semicolon. Invocation line controls are masters and override the same controls in the program source. Examples of assembler control capabilities are: format control of the assembly listing, enable/disable listing of conditional code and conditional directives, listing of comment lines, macro expansion lines, macro object lines only. Cross references and symbol tables can be generated in the listing file, macro local symbols and constants can be put into the symbol table, number of assembler passes specified, assembler controls saved and restored . . .

ASSEMBLER INVOCATION

ASMHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for ASMHPC invocation are: the name of the assembly program(s) or module(s) to be assembled, list of assembler options and the name of a command file that contains additional invocation line source filenames and/or options. An assembler invocation line option is an assembler control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the ASMHPC HELP menu is displayed.

HPC Cross-Linker—LNHPC

INTRODUCTION

The MOLE HPC cross-linker (LNHPC) links object files generated by ASMHPC. The result is an absolute load module in various formats, such as the MOLE ".lm" format, INTEL Hex or COFF formats. LNHPC combines a number of ASMHPC relocatable object modules into a single absolute object module with all the relocatable addresses assigned. All external symbol references between modules are resolved, and library object modules are linked as required.

LNHPC creates two outputs:

1. An absolute object module file that can be downloaded to the MOLE development system for emulation and debugging. The output could also be used by the HPC Source Level Debugger if the SYMBOL option was used on CCHPC to create symbolic information.
2. A load map that shows the result of the link with an optional cross reference listing.

LNHPC MEMORY ALLOCATION

The Linker places each section in memory based on the attributes of the section and the memory that is available. Available memory is specified by the RANGE command. Each section has the following attributes:

Memory type—BASE, ROM8, ROM16, RAM8, RAM16

Size—determined from the object modules

Absolute—section was specified as absolute in assembler

Fixed—starting address was specified by the SECT command

Ranged—memory range was specified by the SECT command.

Memory is allocated section by section. Sections are allocated in the following order:

1. Each absolute or fixed section is placed in memory at its specified address.
2. Each ranged section is placed in memory within the specified range, regardless of whether this memory has been allocated in the Range Definition. An error will occur if the section can not be located.
3. All remaining sections are allocated as follows: As each section is processed, the ranges for its memory type are examined to find enough free space to allocate the section. Each range is examined in order. The first space large enough to contain the section is used. At this point, the memory allocated is marked used. If not enough memory is available to allocate the section, an error message is displayed. For efficiency, sections which may contain word aligned data (ROM16, RAM16, BASE which are word aligned) are allocated first. The user will benefit if the word aligned data is placed in these sections and byte data in other sections.

The load map shows the following:

— Range definitions showing the memory ranges specified by the /RANGE option or by the default.

— The Memory Order Map showing the starting and ending addresses of each contiguous range of memory used.

— The Memory Type Map showing how memory is allocated organized by the memory type.

— The Total Memory Map showing the allocation of all ROM and all RAM.

— The Section Table showing each section in the link, along with its starting and ending address. Section attributes are also displayed.

LINKER INVOCATION

LNHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for LNHPC invocation are—the name of the object file(s), module(s) or libraries to be linked, list of linker options and the name of a command file that contains additional invocation line source filenames and/or options. A linker invocation line option is a linker control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the LNHPC HELP menu is displayed.

HPC Cross-Librarian—LIBHPC

INTRODUCTION

The MOLE HPC cross-librarian (LIBHPC) reads object modules produced by ASMHPC and combines them into one file called a library. The linker can then search the library for any undefined external symbols and link the object module associated with the external symbol. LNHPC will only link in those library object modules required to satisfy external references to maximize efficient use of memory space. LIBHPC is a librarian utility that is provided to allow the user to develop standard modules and place them in libraries. The user may add, delete and list modules in a library file. A library of typical C functions is supplied with the HPC C Compiler (CCHPC). This library is an example of the type of library that could be created for an HPC application program. It is intended to be used as a template for the user to create a custom library specific to the application for maximum code efficiency.

HPC Cross-Librarian—LIBHPC (Continued)

LIBRARIAN INVOCATION

LIBHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for LIBHPC invocation are—the name of the library file to process, list of librarian options and the name of a command file that contains additional invocation line source filenames and/or options. A librarian invocation line option is a librarian control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the LIBHPC HELP menu is displayed.

HPC Source Debugger—DBHPC

The HPC Source Debugger is designed to be a source-level debugger for the HPC family of microcontrollers. This package will have capabilities similar to the HPC MOLE, yet offer the support of symbolics and source code debugging facilities for C language programs. DBHPC will execute on the IBM PC or compatible machines and control the MOLE development station using interactive sequences transparent to the user to effect high level debug functions. DBHPC will also provide download capability to the MOLE.

DBHPC will have the ability to display and modify data as symbolic variables or as variable addressing expressions in the context of the program. Data will be displayed as the variable or expression type, but can be overridden by specifiers derived from "printf"-type controls. The HPC hardware registers are available for display and modification. Those registers defined and used by the program will be available in the context of their use by the program and with the others, e.g. interrupt or processor registers, will also be available, even though the program does not access them.

The debugger will provide hardware Breakpoint capability as supplied by the MOLE. Also available will be software qualification of Breakpoints done by the PC to provide the user with the capability of specifying complex Breakpoint triggering conditions. Breakpoints can be set for execution of functions, source lines or addresses and accesses to variables or data addresses. Singlestepping will be available by source code line or by machine instruction. Functions may be stepped through as if it were a single operation or into the function to examine the internal actions within the function. During Breakpoint, the trace will record and display the prior 2000 HPC accesses, machine instructions or C source lines.

DBHPC will display source code and search for strings in the source code. The display of source code can be C source from the file, disassembled assembly code, or as C source with corresponding assembly code intermixed. The C stack can be displayed showing the current function calling history. The program variables and data may be autoprinted on execution of a Breakpoint.

A history file may be created with DBHPC recording the DBHPC input and responses. The history file, or a simple ASCII file, may be used as input to DBHPC. This provides the capability of re-running debug sessions. Operating system commands can be executed from within DBHPC and control can be passed back to the debugger. A transparent mode will allow interaction through the debugger directly to the MOLE.

DBHPC uses COFF files generated by the HPC C Compiler, Assembler, and Linker to obtain the object code and symbolic information required for debugging. The COFF files are created with the /symbol switch on the CCHPC and ASMHPC packages.

Following is a summary of the commands and features of DBHPC:

Commands for Data Manipulation

Command	Options
	Byte
Alter	Word
Deposit	Long
Find	Float
List	Pointer
	Chars
Type	
Modify	

These MOLE commands are available on the HPC Debugger with additional arguments. The Byte, Word, . . . options are used to define the data type to be used in performing this operation.

The Type and Modify commands have only Byte, Word and Long options.

Added to the MOLE commands for data manipulation is a command for displaying variables, values of expressions and data pointed at by variables or address expressions. This command is called View.

HPC Source Debugger—DBHPC (Continued)

Debug Commands

Autoprint	Displays user selected information on each Breakpoint, Watchpoint or Singlestep.
Breakpoint	MOLE hardware Breakpoint specified with C source trigger conditions.
Clear	Clears Breakpoint, Time, Trace, Watchpoint enables.
Watchpoint	Hardware/Software Breakpoint allowing the user to define complex triggering conditions and sequences. The user program will not execute in real time until reaching the Watchpoint condition in all cases.
Trace	MOLE hardware Trace specified with C source trigger conditions.
Time	MOLE Time function specified with C source trigger conditions.
Singlestep	Step each C source line.
Next Source	Step over a function.
Single Inst.	Step each machine instruction. This is the MOLE Singlestep function.
Next Inst.	Step over subroutines. This is the MOLE Next function.
Reset	Resets the HPC on the MOLE.
RGo	Performs a Reset and Go command.
Go	Starts running the HPC or enables a Breakpoint, Trace, Time or Watchpoint command.
Search	Search Trace memory for specified occurrence.
Stack History	Display program stack showing functions, arguments and local variables.
List Source	Lists C source code.
List Xtended	Lists program as C source followed by assembly code.
Put	MOLE Put function allows input of assembly code line-by-line.

Commands for Controlling Status

Radix	Sets default radix for input and display.
Trace Mode	Sets mode of Trace. Capture of source line, machine cycle or machine instruction.
Status	Display chip and debugger status.
Help	Displays Help menu.
End	Ends debugger session.
History	Create a History file on disk.

Special Commands

Load	Load file from disk to MOLE and initialize DBHPC.
Map	Map emulation memory on or off-board the MOLE.
Bypass	Bypass debugger and communication directly with MOLE.
Chip	Define chip and system memory configuration for MOLE.
Diagnostic	Run MOLE on-board Diagnostics.
!	Invoke a shell or execute a DOS command.

HPC Source Debugger—DBHPC (Continued)

HOW TO ORDER HPC SOFTWARE

HPC software is available for a variety of host environments. To order a software package, select the host system and order the part number listed.

Included, along with the cross assembler, in the software package are two file conversion routines to convert the assembler output (LM) to HEX and to convert HEX to LM. Also included in the software package is a COMM program which facilitates the downloading and uploading between the host and the MOLE, and adds the capability to make the host act as a terminal.

The C compiler package also includes the relocatable assembler. Order one or the other but not both.

An HPC software evaluation package is available (MOLE-HPC-IBMEVAL) that will allow up to 1000 lines of code to be compiled, assembled and linked.

Software Selection Table

Host*	Order Part Number	Description	Includes	Manual Number
IBM-PC	MOLE-HPC-IBMR	Relocatable Assembler Software for IBM (ASMHPC, LIBHPC, LNHPC, DBHPC)	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM (CCHPC)	HPC C Compilers Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001 424410897-001
	MOLE-HPC-IBMEVL	Evaluation Module	Includes Assembler and C Compiler Evaluation Software	

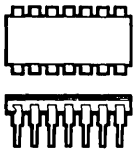
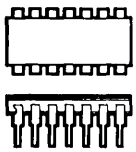
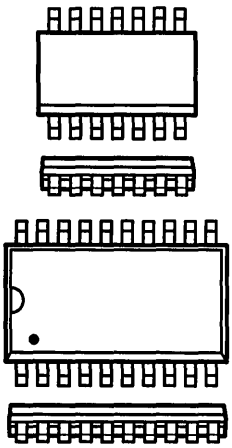
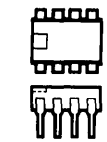
*VAX, VMS and VAX UNIX will be supported in the near future. Contact field sales for more information.

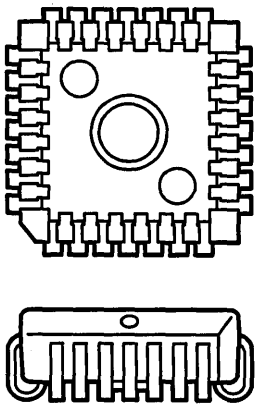
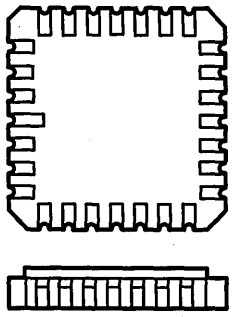
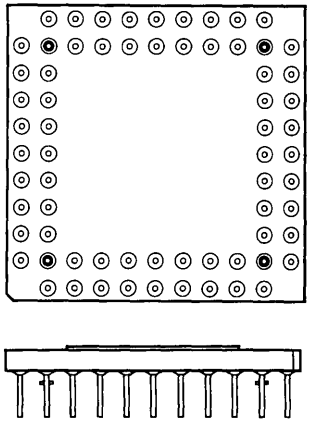
Section 9

Appendices/ Physical Dimensions

Section 9 Contents

Industry Package Cross-Reference Guide

	NSC	Signetics	Intel	Motorola	TI	RCA	Hitachi	NEC
 8-, 14-, 16-, 20- and 28-Lead Glass/Metal DIP	D	I	C	L		D	C	D
 8-, 14- 16-, 20-, 24- and 28-Lead Low Temperature Ceramic DIP	J	F	D	U	J		G	D
 SO (Narrow Body)	M	D		D	D	M	MP	G
(Wide Body)	WM				DW			
 8-, 14- 16-, 20- 24- and 28-Lead Plastic DIP	N	V, A, B	P	P	P, N	E	P	C

	NSC	Signetics	Intel	Motorola	TI	RCA	Hitachi	NEC
 PCC	V	A	N	FN	FN	Q	CP	L
 LCC Leadless Ceramic Chip Carrier	E	G	R	U	FK/ FG/FH	BJ	CG	K
 PGA Ceramic Pin Grid Array	U		CG					

National Semiconductor

Transmission Line Drivers/Receivers

The common purpose of transmission line drivers and receivers is to transmit data quickly and reliably through a variety of environments over electrically long distances. This task is complicated by the fact that externally introduced noise and ground shifts can severely degrade the data.

The connection between two elements in a system should be considered a transmission line if the transmitted signal takes longer than twice its rise or fall time to travel from the driver to the receiver.

Single-Ended Data Transmission

In data processing systems today there are two basic means of communicating between components. One method is single-ended, which uses only one signal line for data transmission, and the other is differential, which uses two signal lines.

The Electronics Industry Association (EIA) has developed several standards to simplify the interface in data communications systems.

RS-232

The first of these, RS-232, was introduced in 1962 and has been widely used throughout the industry. RS-232 was developed for single-ended data transmission at relatively slow data rates (20 kBaud) over short distances (up to 50 ft.).

RS-423

With the need to transmit data faster and over longer distances, RS-423, a newer standard for single-ended applications, was established. RS-423 extends the maximum data rate to 100 kBaud (up to 30 ft.) and the maximum distance to 4000 feet (up to 1 kBaud). RS-423 also requires high impedance driver outputs with power off so as not to load the transmission line.

Differential Data Transmission

When transmitting at very high data rates, over long distances and through noisy environments, single-ended transmission is often inadequate. In these applications, differential data transmission offers superior performance. Differential transmission nullifies the effects of ground shifts and noise signals which appear as common mode voltages on the transmission line.

RS-422

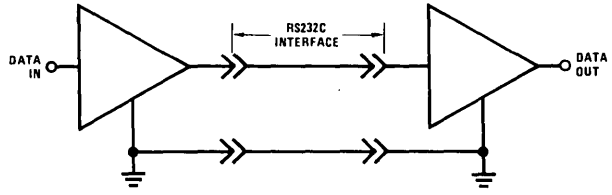
RS-422 was defined by the EIA for this purpose and allows data rates up to 10 MBaud (up to 40 ft.) and line lengths up to 4000 feet (up to 100 kBaud).

Drivers designed to meet this standard are well suited for party-line type applications where only one driver is connected to, and transmits on, a bus and up to 10 receivers can receive the data. While a party-line type of application has many uses, RS-422 devices cannot be used to construct a truly multipoint bus. A multipoint bus consists of multiple drivers and receivers connected to a single bus, and any one of them can transmit or receive data.

RS-485

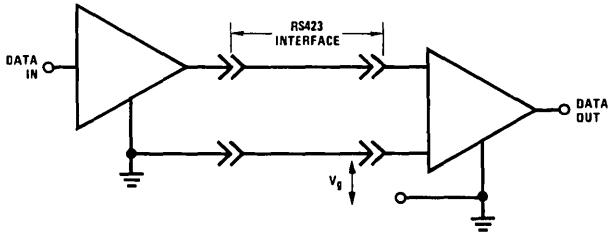
To meet the need for truly multipoint communications, the EIA established RS-485 in 1983. RS-485 meets all the requirements of RS-422, but in addition, this new standard allows up to 32 drivers and 32 receivers to be connected to a single bus—thus allowing a truly multipoint bus to be constructed.

RS-232C Application



TL/00/2901-1

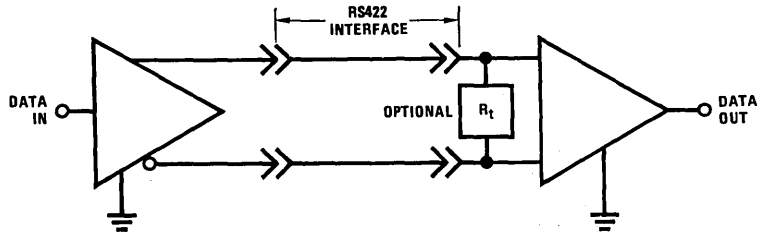
EIA RS-423 Application



TL/00/2901-2

Differential Data Transmission (Continued)

EIA RS-422 Application



TL/00/2901–3

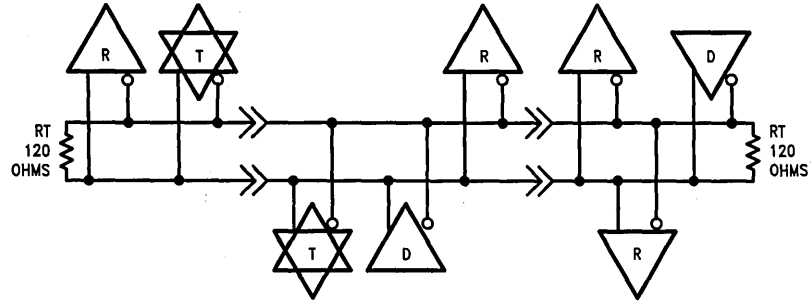
The key features of RS-485:

- Implements a truly multipoint bus consisting of up to 32 drivers and 32 receivers
- An extended common-mode range for both drivers and receivers in TRI-STATE and with power off (−7V to +12V)
- Drivers can withstand bus contention and bus faults

National Semiconductor produces a variety of drivers, receivers, and transceivers for these four very popular transmission standards and numerous other data transmission requirements.

Shown below is a table that highlights key aspects of the EIA Standards. More detailed comparisons can be found in the various application notes in Section 1.

RS-485 Application



D — Driver
R — Receiver
T — Transceiver

TL/00/2901–4

Specification		RS-232C	RS-423	RS-422	RS-485
Mode of Operation		Single-Ended	Single-Ended	Differential	Differential
Number of Drivers and Receivers Allowed on One Line		1 Driver, 1 Receiver	1 Driver, 10 Receivers	1 Driver, 10 Receivers	32 Drivers, 32 Receivers
Maximum Cable Length		50 feet	4000 feet	4000 feet	4000 feet
Maximum Data Rate		20 kb/s	100 kb/s	10 Mb/s	10 Mb/s
Driver Output Maximum Voltage		±25V	±6V	−0.25V to +6V	−7V to +12V
Driver Output Signal Level	Loaded	±5V	±3.6V	±2V	±1.5V
	Unloaded	±15V	±6V	±5V	±5V
Driver Load Impedance		3 kΩ to 7 kΩ	450Ω min	100Ω	54Ω
Maximum Driver Output Current (High Impedance State)	Power On	——	——	——	±100 μA
	Power Off	V_{MAX}/300Ω	±100 μA	±100 μA	±100 μA
Slew Rate		30 V/μs max	Controls Provided	——	——
Receiver Input Voltage Range		±15V	±12V	−7V to +7V	−7V to +12V
Receiver Input Sensitivity		±3V	±200 mV	±200 mV	±200 mV
Receiver Input Resistance		3 kΩ to 7 kΩ	4 kΩ min	4 kΩ min	12 kΩ min

National Semiconductor

Plastic Leaded Chip Carrier (PLCC) Packaging

General Description

The Plastic Leaded Chip Carrier (PLCC) is a miniaturized low cost semiconductor package designed to replace the Plastic Dual-In-Line Package (P-DIP) in high density applications. The PLCC utilizes a smaller lead-to-lead spacing—0.050" versus 0.100" - and leads on all four sides to achieve a significant footprint reduction over the P-DIP. The rolled under J-bend leadform separates this package style from other plastic quad packages with flat or gull wing lead forms. As with virtually all packages of 0.050" or less lead spacing, the PLCC requires surface mounting to printed circuit boards as opposed to the more conventional thru-hole mounting of the P-DIP.

History

The Plastic Leaded Chip Carrier with J-bend leadform was first introduced in 1976 as a premolded plastic package. The premolded version has yet to become popular but the quad format with J-Bend leads has been adapted to traditional post molded packaging technology (the same technology used to manufacture the P-DIP). In 1980 National Semiconductor developed a post molded version of the PLCC. The J-bend leadform allowed them to adopt the footprint connection pattern already registered with JEDEC for the leadless chip carrier (LCC). In 1981 a task force was organized within JEDEC to develop a PLCC registration for package I/O counts of 20, 28, 44, 52, 68, 84, 100, and 124. A registered outline was completed in 1984 (JEDEC Outline MO-047) after many changes and improvements over the original proposals. This first PLCC registration covers square packages with an equal number of leads on all sides. A second registration, MO-052, was completed in 1985 for rectangular packages with I/O counts of 18, 22, 28 and 32.

Since 1980 many additional semiconductor manufacturers and packaging subcontractors have developed PLCC capability. There are now well over 20 sources with the number growing steadily.

Surface Mounting

Surface mounting refers to component attachment whereby the component leads or pads rest on the surface of the PCB instead of the traditional approach of inserting the leads into through-holes which go through the board. With surface mounting there are solder pads on the PCB which align with the leads or pads on the component. The resulting solder joint forms both the mechanical and electrical connection.

ADVANTAGES

The primary reason for surface mounting is to allow leads to be placed closer together than the 0.100" standard for DIPs with through-hole mounting. Through-hole mounting on smaller than 0.100" spacing is difficult to achieve in production and generally avoided. The move to 0.050" lead spacing offered with the current generation of surface mounted components, along with a switch from a dual-in-line format to a quad format, has achieved a threefold increase in component mounting density. A need to achieve greater density is a major driving force in today's marketplace.

MANUFACTURING TECHNIQUES

Learning how to surface mount components to printed circuit boards requires the user to become educated in new assembly processes not typically associated with through-hole insertion/wave soldering assembly methods.

Surface mounting involves three basic process steps:

1) Application of solder or solder paste to the printed circuit board.
2) Positioning of the component onto the printed circuit board
3) Reflowing of the solder or solder paste.

As with any process, there are many details involved to achieve acceptable throughput and acceptable quality. National Semiconductor offers a surface mounting guide which deals with the specifics of successful surface mounting. We encourage the user to review this document and to contact us if further information on surface mounting is desired.

Benefits of the PLCC

There are four principle advantages offered the user by switching from P-DIP to PLCC. These four advantages are outlined below as follows:

1. Increased Density—
   - — Typically 3-to-1 size reduction of printed circuit boards. See *Figure 1* for a footprint comparison between PLCC and P-DIP. This can be as high as 6-to-1 in certain applications.
   - — Surface mounting allows components to be placed on both sides of the board.
   - — Surface mount and thru-hole mount components can be placed on the same board.
   - — The large diameter thru-holes can be reduced in number, entirely eliminated, or reduced in size (if needed for via connection).
2. Increased Performance—
   - — Shorter traces on printed circuit boards.
   - — Better high frequency operation.
   - — Shorter leads in package. *Figure 2* and Table I compare PLCC and P-DIP mechanical and electrical characteristics.
3. Increased Reliability—
   - — Leads are well protected.
   - — Fewer connectors.
   - — Simplified rework.
   - — Vibration and shock resistant.
4. Reduced Cost—
   - — Fewer or smaller printed circuit boards.
   - — Less hardware.
   - — Same low cost printed circuit board material.
   - — Plastic packaging material.
   - — Reduced number of costly plated-through-holes.
   - — Fewer circuit layers.

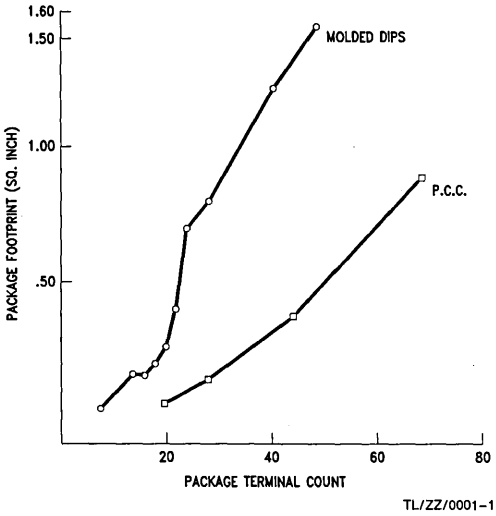


FIGURE 1. Footprint Area of PLCC vs. P-DIP

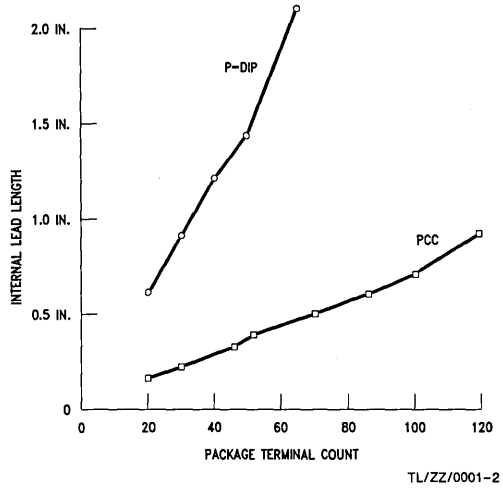


FIGURE 2. Longest Internal Lead PLCC vs. P-DIP

TABLE I. Electrical Performance of PLCC vs. P-DIP (44 I/O PLCC vs. 40 I/O P-DIP, both with Copper Leads)

Criteria	Shortest Lead		Longest Lead	
	PLCC	P-DIP	PLCC	P-DIP
Lead Resistance (Measured)	3Ω	4Ω	6Ω	7Ω
Lead-to-Lead Capacitance (Measured on Adjacent Leads)	0.1 pF	0.1 pF	0.3 pF	3.0 pF
Lead Self-Inductance (Calculated)	3.2 nH	1.4 nH	3.5 nH	19.1 nH

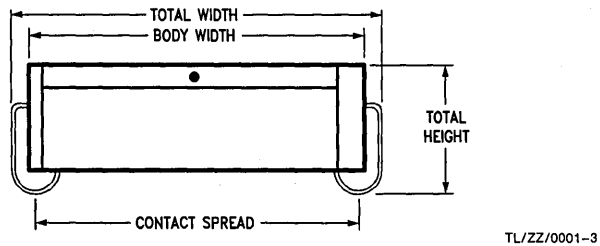


FIGURE 3. Package Outline

TABLE II. Principle Dimensions Inches/(Millimeters) (Refer to *Figure 3*)

Lead Count	Total Width		Total Height		Body Width		Contact Spread	
	Min	Max	Min	Max	Min	Max	Min	Max
20	0.385 sq. (9.779)	0.395 sq. (10.03)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.345 sq. (8.763)	0.355 sq. (9.017)	0.310 sq. (7.874)	0.330 sq. (8.382)
28	0.485 sq. (12.32)	0.495 sq. (12.57)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.445 sq. (11.30)	0.455 sq. (11.56)	0.410 sq. (10.41)	0.430 sq. (10.92)
44	0.685 sq. (17.40)	0.695 sq. (17.65)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.645 sq. (16.38)	0.655 sq. (16.64)	0.610 sq. (15.49)	0.630 sq. (16.00)

TABLE II. Principle Dimensions Inches/(Millimeters) (Refer to *Figure 3*) (Continued)

Lead Count	Total Width		Total Height		Body Width		Contact Spread	
	Min	Max	Min	Max	Min	Max	Min	Max
68	0.985 sq. (25.02)	0.995 sq. (25.27)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.945 sq. (24.00)	0.955 sq. (24.26)	0.910 sq. (23.11)	0.930 sq. (23.62)
84	1.185 sq. (30.10)	1.195 sq. (30.36)	0.165 sq. (4.191)	0.180 sq. (4.572)	1.150 sq. (29.21)	1.158 sq. (29.41)	1.110 sq. (28.20)	1.130 sq. (28.70)
124	1.685 sq. (49.13)	1.695 sq. (49.39)	0.180 sq. (4.572)	0.200 sq. (5.080)	1.650 sq. (41.91)	1.658 sq. (42.11)	1.610 sq. (40.90)	1.630 sq. (41.40)

TABLE III. Package Thermal Resistance
(Deg. C/Watt, Junction-to-Ambient, Board Mount)

Lead Count	Device Size		
	1,000 Mil^2	10,000 Mil^2	100,000 Mil^2
20	102	85	67
28	95	73	55
44	54	47	40
68	44	40	38
84*	40	35	30
124*	40	35	30

*Estimated values

Package Design Criteria

Experience has taught us there are certain criteria to the PLCC design which must be followed to provide the user with the proper mechanical and thermal performance. These requirements should be carefully reviewed by the user when selecting suppliers for devices in PLCC. Some of these are covered by the JEDEC registration and some are not. These important requirements are listed in Table IV.

Reliability

National Semiconductor utilizes an assembly process for the PLCC which is similar to our P-DIP assembly process. We also utilize identical materials. This is a very important point when considering reliability. Many years of research and development have gone into steadily improving our P-DIP quality and maintaining a leadership position in plastic package reliability. All of this technology can be directly applied to the PLCC. Table V shows the results of applying this technology to the PLCC. As we make further advances in plastic package reliability, these will also be applied to the PLCC.

Sockets

There are several manufacturers currently offering sockets for the plastic chip carrier. Following is a listing of those manufacturers. The listing is divided into test/burn-in and production categories. There may be some individual sockets that will cover both requirements.

TABLE IV. Package Design Criteria

Criteria	Required to Comply with JEDEC Registration
Minimum Inside Bend Radius of Lead at Shoulder Equal or Greater than Lead Thickness—to Prevent Lead Cracking/Fatigue	Not Required
Minimum One Mil Clearance Between Lead and Plastic Body at all Points—to Provide Lead Compliancy and Prevent Shoulder Joint Cracking/Fatigue	Not Required
Copper Leads for Low Thermal Resistance	Not Required
Minimum 10 Mil Lead Thickness for Low Thermal Resistance and Good Handling Properties	Not Required
Minimum 26 Mil Lead Shoulder Width to Prevent Interlocking of Devices During Handling	Yes
Maximum 4 Mils coplanarity Across Seating Plane of all Leads	Yes

TABLE V. Reliability Test Data
(Expressed as Failures per Units Tested)

Device/Package	OPL	TMCL	TMSK	BHTL	ACLV
LM324/20 Lead	0/96	0/199	0/50	0/97	0/300
LF353/20 Lead	0/50	0/50	—	0/45	0/100
DS75451/20 Lead	0/47	—	0/50	0/93	0/179
DM875191/28 Lead	0/154	0/154	0/154	0/154	0/154
DM875181/28 Lead	0/77	0/77	0/77	0/77	0/77

OPL = Dynamic high temperature operating life at 125°C or 150°C, 1,000 hours.
TMCL = Temperature cycle, Air-to-Air, −40°C to +125°C or −65°C to +150°C, 2,000 cycles.
TMSK = Thermal shock, Liquid-to-Liquid, −65°C to +150°C, 100 cycles.
BHTL = Biased humidity temperature life, 85°C, 85% humidity, 1,000 hours.
ACLV = Autoclave, 15 psi, 121°C, 100% humidity, 1,000 hours.

Production Sockets

AMP
Harrisburg, PA
(715) 564-0100

Augat
Attleboro, MA
(617) 222-2202

Burndy
Norwalk, CT
(203) 838-4444

Methode
Rolling Meadows, IL
(312) 392-3500

Textool
Irving, TX
(214) 259-2676

Thomas & Betts
Raritan, NJ
(201) 469-4000

Test/Burn-In Sockets

Plastronics
Irving, TX
(214) 258-1906

Textool
Irving, TX
(214) 259-2676

Yamaichi
c/o Nepenthe Dist.
(415) 856-9332

ADDITIONAL INFORMATION AND SERVICES

National Semiconductor offers additional Databooks which cover surface mount technology in much greater detail. We also have a surface mount laboratory to provide demonstrations and customer support, as well as technology development. Feel free to contact us about these additional resources.

National Semiconductor

TapePak®

The latest generation in VLSI packaging, TapePak is the package of the future—low-cost, reliable, high-leadcount packaging that's easy to handle, easy to test, and easy to mount. It's also compatible with existing surface-mount technology.

TapePak uses tape-automated bonding technology and a unique outer ring (patent pending) to protect the leads and, at the same time, provide an effective test interface.

This outer ring is molded at the same time as the body of the package and creates test points outside the package leads. The test ring is discarded along with the tape as the package is excised by the automatic pick-and-place machine at the point of assembly.

During testing, the leads themselves never come in contact with the test socket, so lead damage and coplanarity problems are eliminated. The test ring also allows burn-in to be performed on each device.

Not only does this ring protect the leads during handling, testing and assembly, but it also allows leads to be placed on 0.020-inch (0.50-mm) centers while the test points are placed on 0.050-inch (1.27-mm) centers. That way, the test points are compatible with existing automatic test equipment.

As a result, packages can be manufactured in smaller sizes with higher leadcounts and still be compatible with automatic assembly systems. With TapePak, packages contain from 40 to more than 300 leads, yet a 300-lead package measures only 1.2 inches (30.5 mm) on a side.

A TapePak device can be less than $1/10$ the size of a traditional DIP and $1/3$ the size of other surface-mount packages such as a PLCC.

TapePak was designed to take full advantage of automatic assembly systems with their high speed and precision. It can be used with existing precision surface-mount assembly equipment with minimal modification. The only requirement is an accessory for removing the test ring and forming the leads at the point of assembly.

TapePak also provides a significant improvement in the electrical characteristics of each package. Lead capacitance and inductance, for example, can be reduced up to ten times that of other packages. Signal propagation time is also reduced, and thermal characteristics are improved.

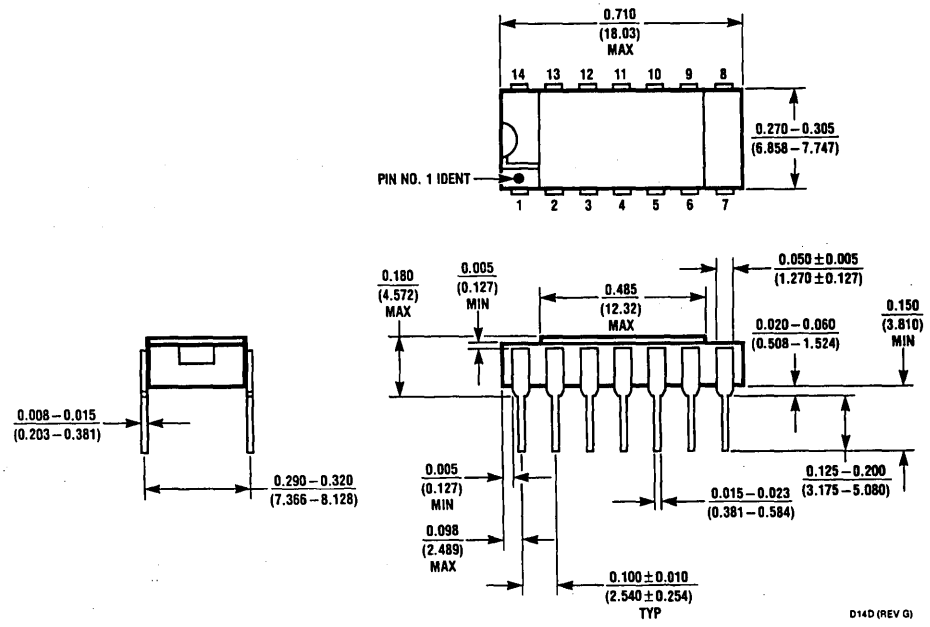
Performance and reliability are improved because there are one-third fewer connections between the die and the PC board. Low-stress molding compounds also improve package reliability. TapePak devices pass stringent environmental tests, including autoclaving at 121°C at 15 psi and thermal shock from −65°C to +150°C for 1000 cycles.

No other package takes similar advantage of materials technology to provide the combination of low cost, high density, testability, damage resistance, and reliability.

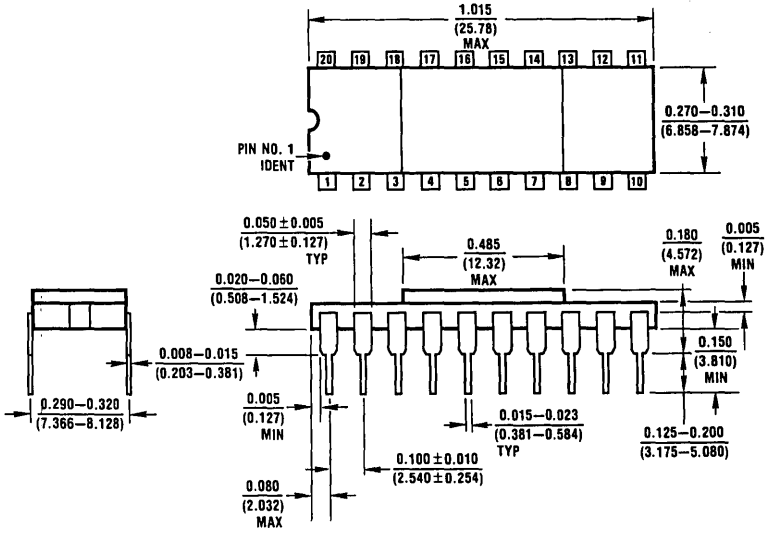
To further the technology in the industry and make these advantages available to everyone, National has submitted TapePak specifications to the JEDEC (Joint Electronic Device and Engineering Council) packaging committee as an industry standard. We have also licensed other manufacturers to use TapePak packaging for their own proprietary devices.

National Semiconductor

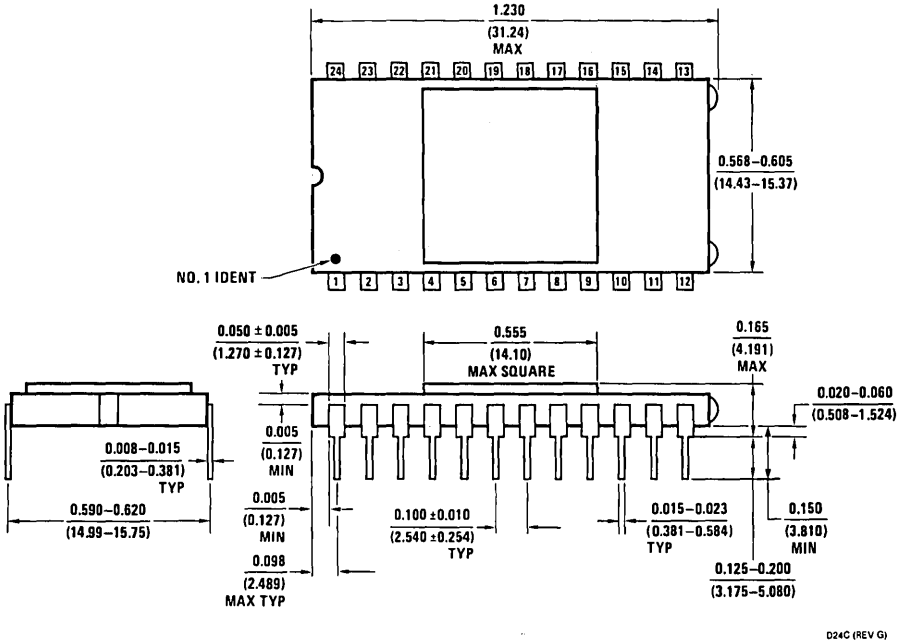
All dimensions are in inches (millimeters)



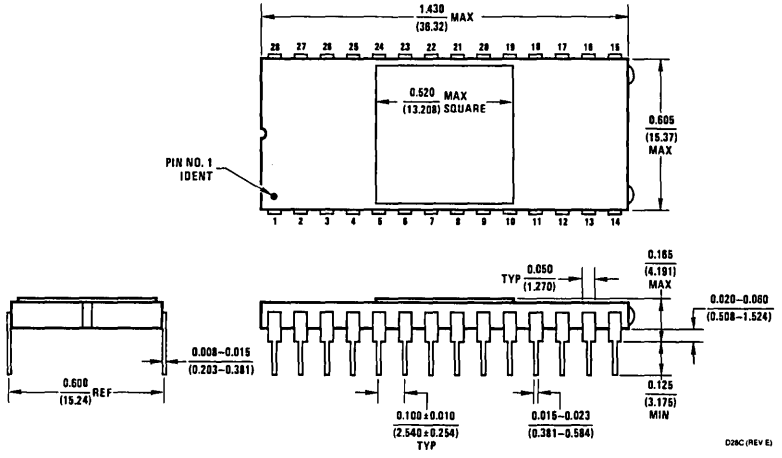
NS Package D14D



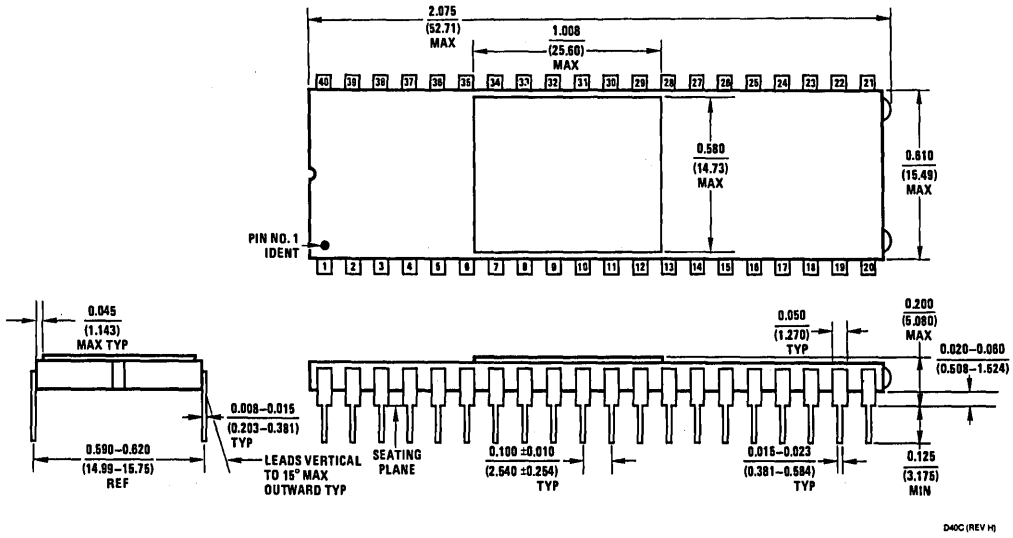
NS Package D20A



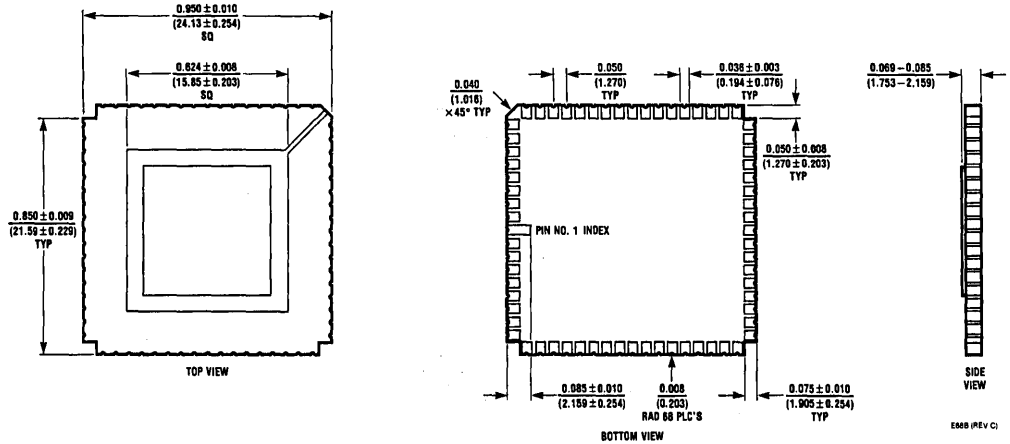
NS Package D24C



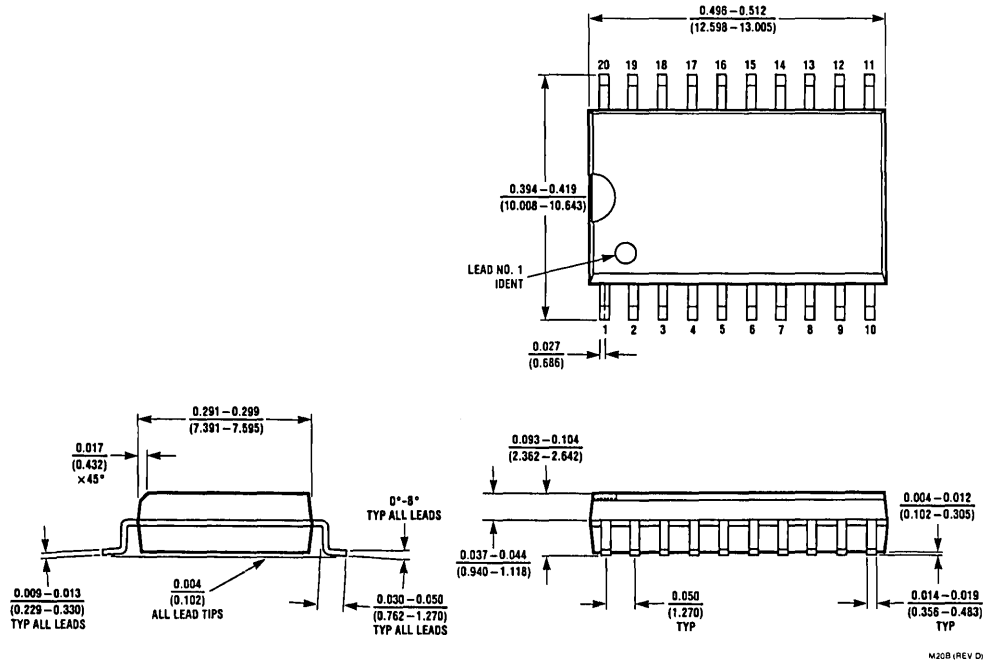
NS Package D28C



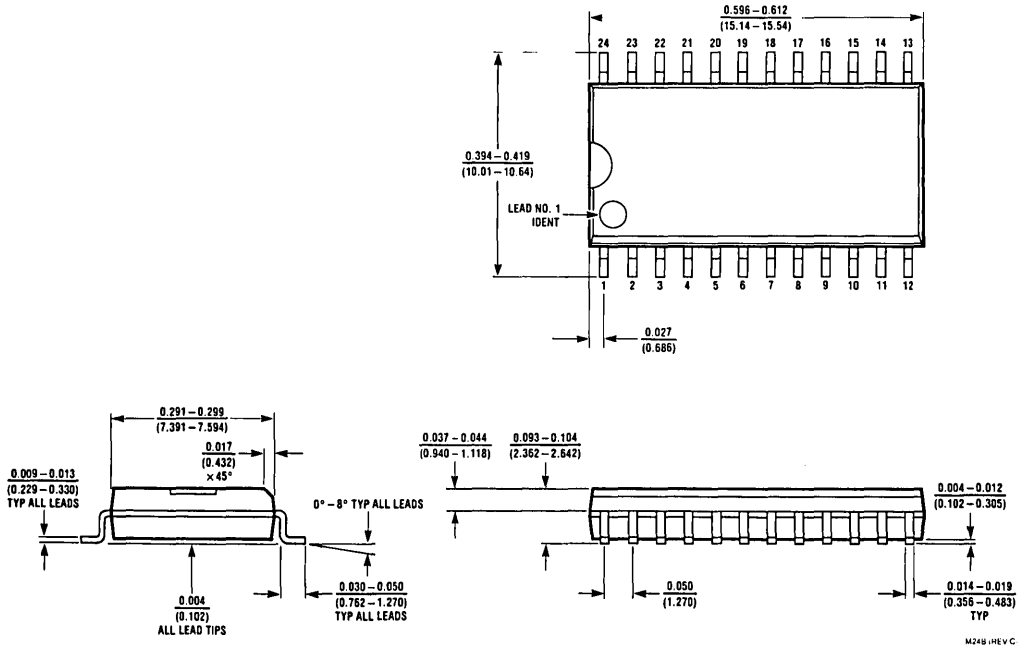
NS Package D40C



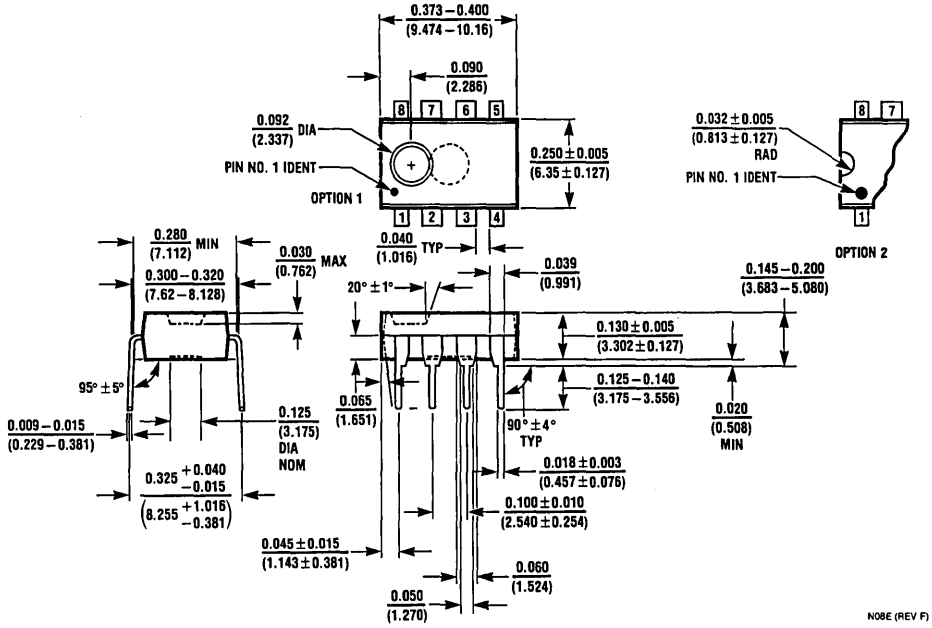
NS Package E68B



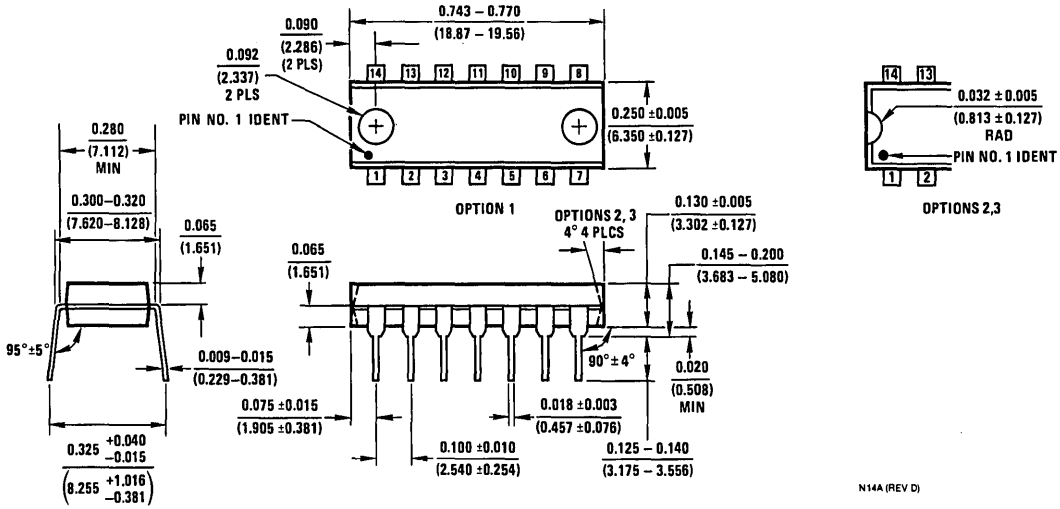
NS Package M20B



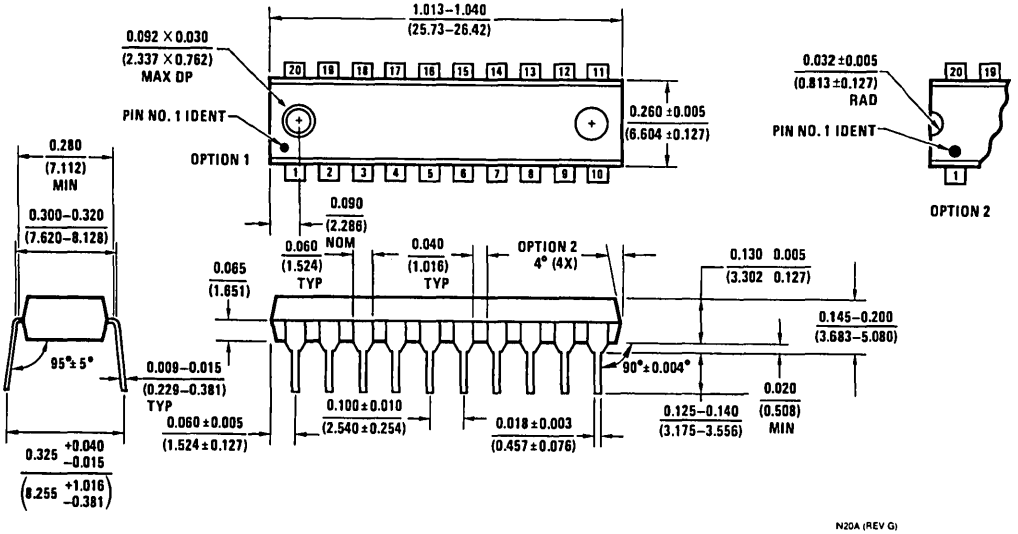
NS Package M24B



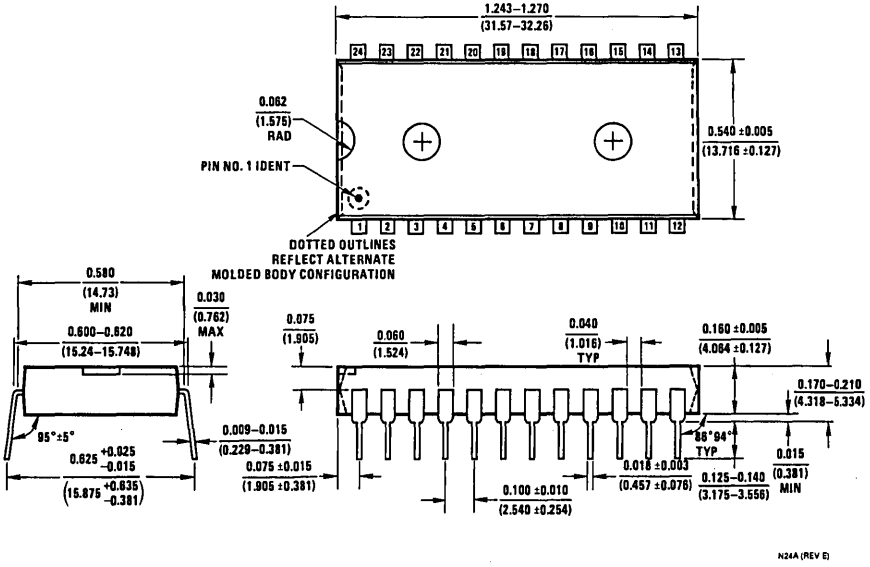
NS Package N08E



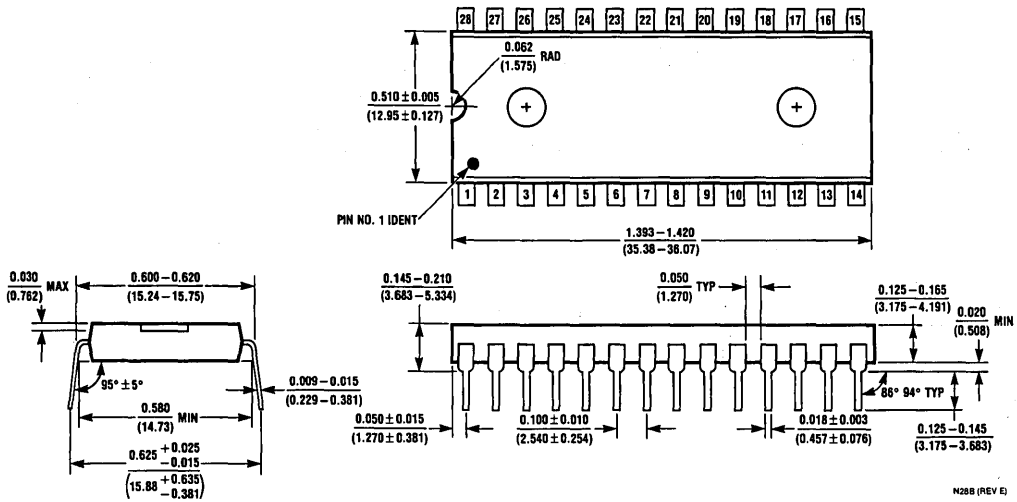
NS Package N14A



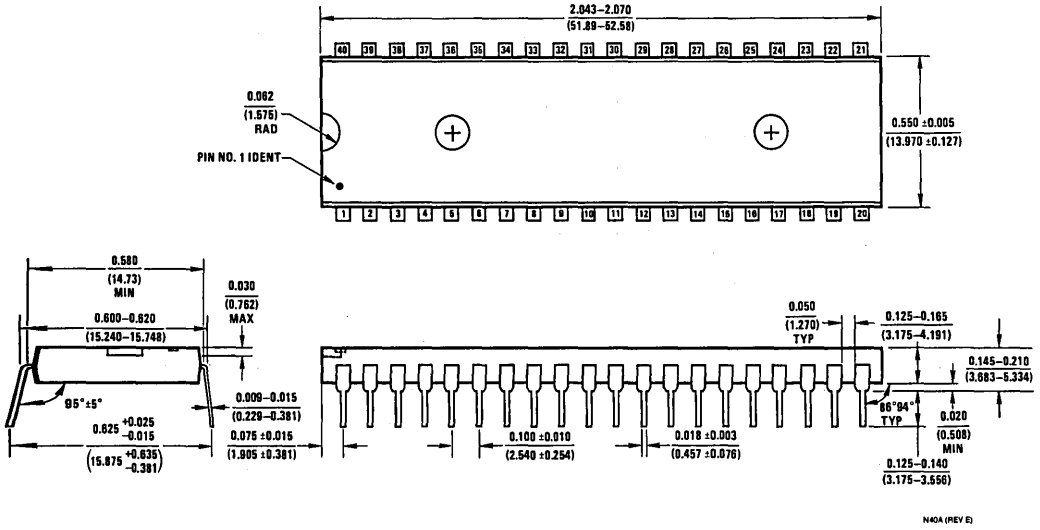
NS Package N20A



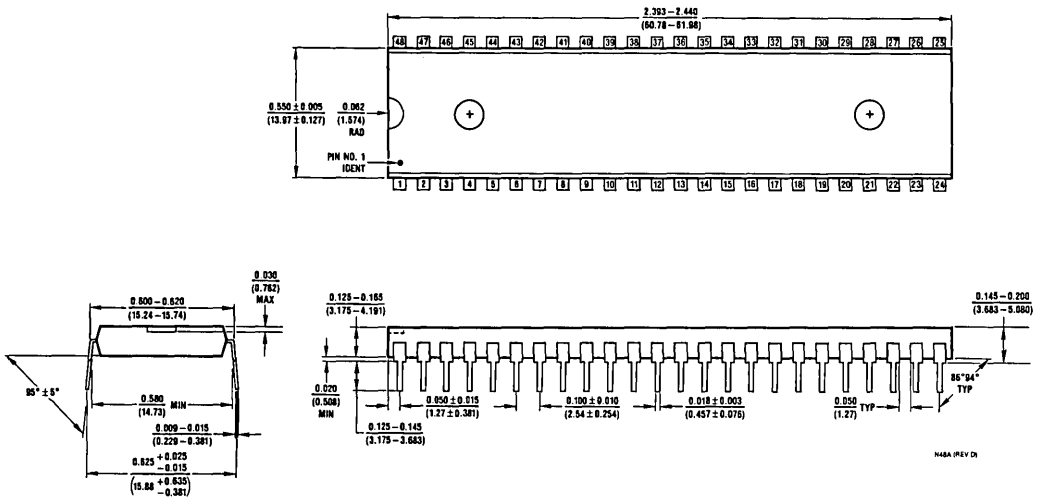
NS Package N24A



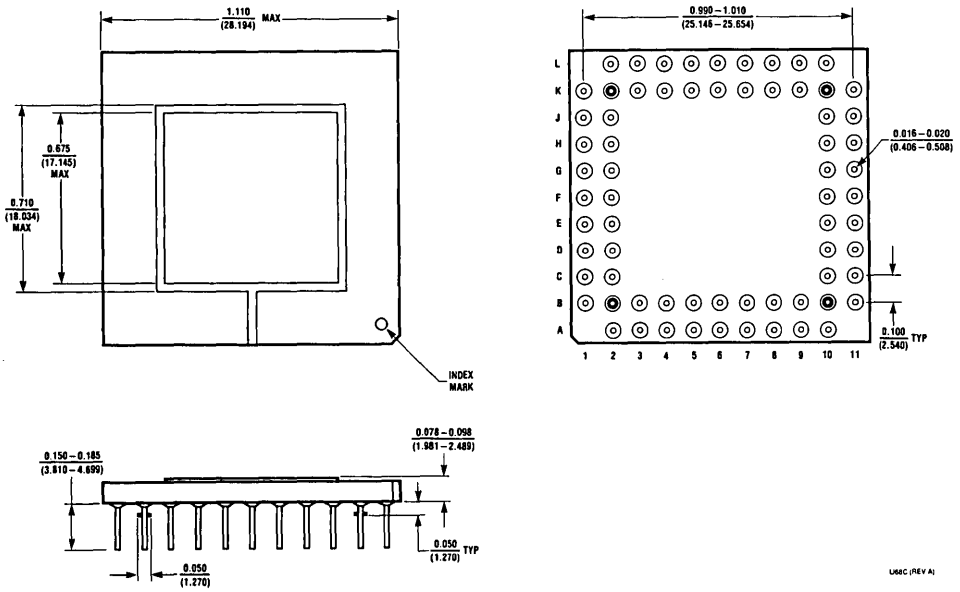
NS Package N28B



NS Package N40A

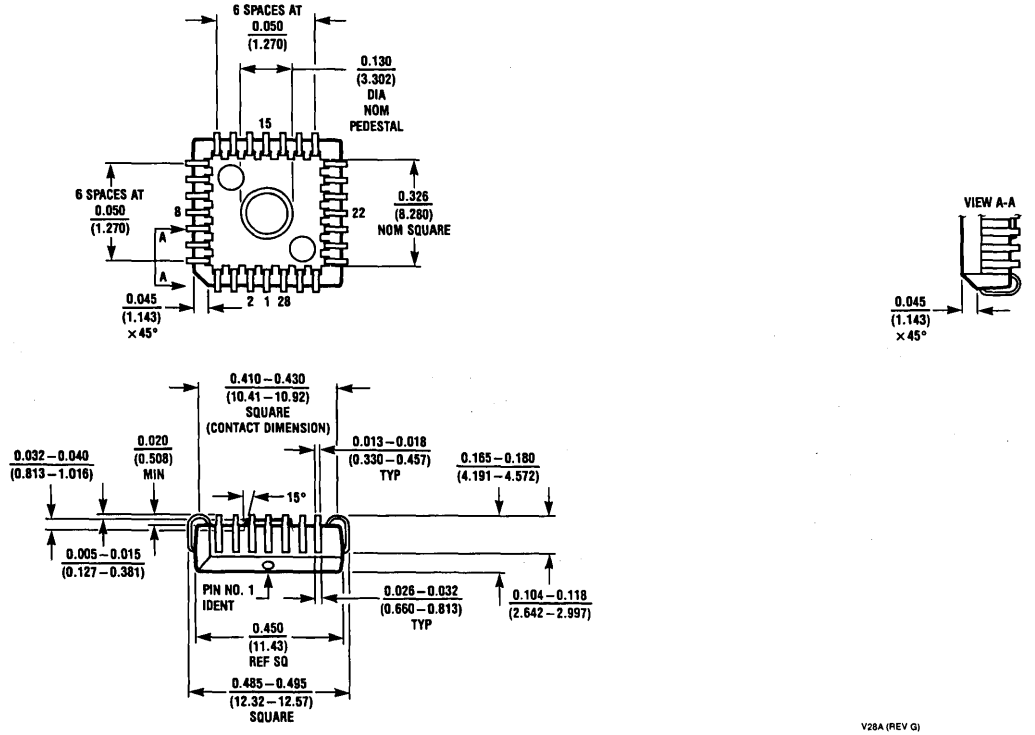


NS Package N48A

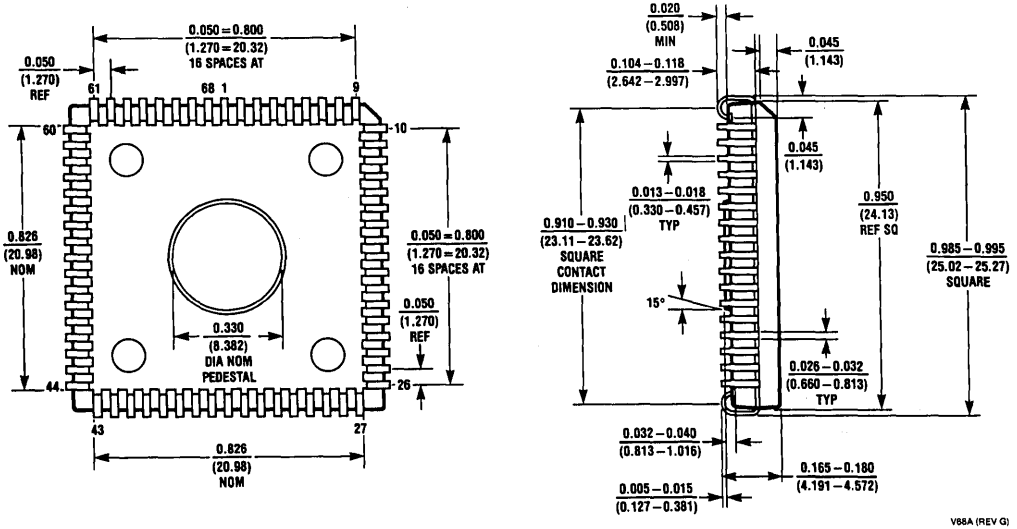


NS Package U68C

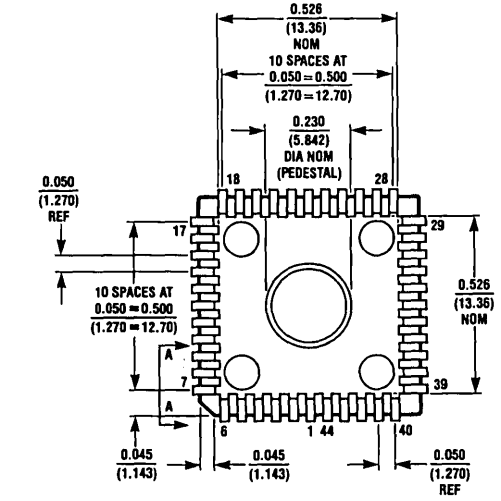
9

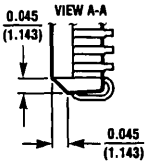


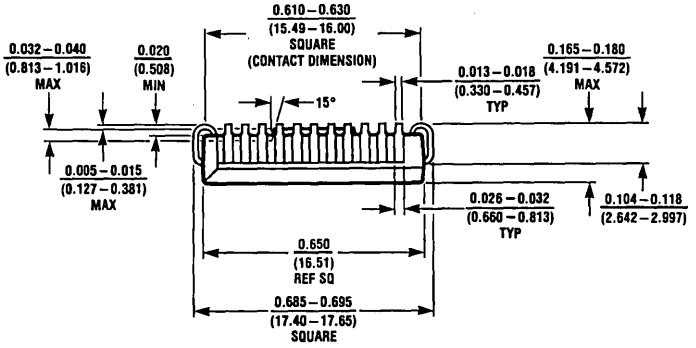
NS Package V28A



NS Package V68A







V44A (REV H)

NS Package V44A

NOTES

NOTES

National Semiconductor

Bookshelf of Technical Support Information

National Semiconductor Corporation recognizes the need to keep you informed about the availability of current technical literature.

This bookshelf is a compilation of books that are currently available. The listing that follows shows the publication year and section contents for each book.

Please contact your local National sales office for possible complimentary copies. A listing of sales offices follows this bookshelf.

We are interested in your comments on our technical literature and your suggestions for improvement.

Please send them to:

Technical Communications Dept. M/S 23-200
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

For a recorded update of this listing plus ordering information for these books from National's Literature Distribution operation, please call (408) 749-7378.

ALS/AS LOGIC DATABOOK—1987

Introduction to Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky

ASIC DESIGN MANUAL/GATE ARRAYS & STANDARD CELLS—1987

SSI/MSI Functions • Peripheral Functions • LSI/VLSI Functions • Design Guidelines • Packaging

CMOS LOGIC DATABOOK—1988

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • Surface Mount

DATA CONVERSION/ACQUISITION DATABOOK—1984

Selection Guides • Active Filters • Amplifiers • Analog Switches • Analog-to-Digital Converters Analog-to-Digital Display (DVM) • Digital-to-Analog Converters • Sample and Hold • Sensors/Transducers Successive Approximation Registers/Comparators • Voltage References

DATA COMMUNICATION/LAN/UART DATABOOK—Rev. 1

LAN IEEE 802.3 • High Speed Serial/IBM Data Communications • ISDN Components • UARTs Modems • Transmission Line Drivers/Receivers

INTERFACE DATABOOK—1988

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral Power Drivers • Display Drivers Memory Support • Microprocessor Support • Level Translators and Buffers • Frequency Synthesis • Hi-Rel Interface

INTERFACE/BIPOLAR LSI/BIPOLAR MEMORY/PROGRAMMABLE LOGIC DATABOOK—1983

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral/Power Drivers Level Translators/Buffers • Display Controllers/Drivers • Memory Support • Dynamic Memory Support Microprocessor Support • Data Communications Support • Disk Support • Frequency Synthesis Interface Appendices • Bipolar PROMs • Bipolar and ECL RAMs • 2900 Family/Bipolar Microprocessor Programmable Logic

INTUITIVE IC CMOS EVOLUTION—1984

Thomas M. Frederiksen's new book targets some of the most significant transitions in semiconductor technology since the change from germanium to silicon. *Intuitive IC CMOS Evolution* highlights the transition in the reduction in defect densities and the development of new circuit topologies. The author's latest book is a vital aid to engineers, and industry observers who need to stay abreast of the semiconductor industry.

INTUITIVE IC OP AMPS—1984

Thomas M. Frederiksen's new book, *Intuitive IC Op Amps,* explores the many uses and applications of different IC op amps. Frederiksen's detailed book differs from others in the way he focuses on the intuitive groundwork in the basic functioning concepts of the op amp. Mr. Frederiksen's latest book is a vital aid to engineers, designers, and industry observers who need to stay abreast of the computer industry.

LINEAR APPLICATIONS HANDBOOK—1986

The purpose of this handbook is to provide a fully indexed and cross-referenced collection of linear integrated circuit applications using both monolithic and hybrid circuits from National Semiconductor.

Individual application notes are normally written to explain the operation and use of one particular device or to detail various methods of accomplishing a given function. The organization of this handbook takes advantage of this innate coherence by keeping each application note intact, arranging them in numerical order, and providing a detailed Subject Index.

LINEAR 1 DATABOOK—1988

Voltage Regulators • Operational Amplifiers • Buffers • Voltage Comparators • Instrumentation Amplifiers • Surface Mount

LINEAR 2 DATABOOK—1988

Active Filters • Analog Switches/Multiplexers • Analog-to-Digital • Digital-to-Analog • Sample and Hold
Sensors • Voltage References • Surface Mount

LINEAR 3 DATABOOK—1988

Audio Circuits • Radio Circuits • Video Circuits • Motion Control • Special Functions • Surface Mount

LINEAR SUPPLEMENT DATABOOK—1984

Amplifiers • Comparators • Voltage Regulators • Voltage References • Converters • Analog Switches
Sample and Hold • Sensors • Filters • Building Blocks • Motor Controllers • Consumer Circuits
Telecommunications Circuits • Speech • Special Analog Functions

LS/S/TTL DATABOOK—1987

Introduction to Bipolar Logic • Low Power Schottky • Schottky • TTL • Low Power

MASS STORAGE HANDBOOK—Rev. 2

Winchester Disk Preamplifiers • Winchester Disk Servo Control • Winchester Disk Pulse Detectors
Winchester Disk Data Separators/Synchronizers and ENDECs • Winchester Disk Data Controller
SCSI Bus Interface Circuits • Floppy Disk Controllers

MEMORY SUPPORT HANDBOOK—1986

Dynamic Memory Control • Error Checking and Correction • Microprocessor Interface and Applications
Memory Drivers and Support

NON-VOLATILE MEMORY DATABOOK—1987

CMOS EPROMs • EEPROMs • Bipolar PROMs

SERIES 32000 DATABOOK—1986

Introduction • CPU-Central Processing Unit • Slave Processors • Peripherals • Data Communications and LAN's
Disk Control and Interface • DRAM Interface • Development Tools • Software Support • Application Notes

RANDOM ACCESS MEMORY DATABOOK—1987

Static RAMs • TTL RAMs • TTL FIFOs • ECL RAMs

RELIABILITY HANDBOOK—1986

Reliability and the Die • Internal Construction • Finished Package • MIL-STD-883 • MIL-M-38510
The Specification Development Process • Reliability and the Hybrid Device • VLSI/VHSIC Devices
Radiation Environment • Electrostatic Discharge • Discrete Device • Standardization
Quality Assurance and Reliability Engineering • Reliability and Documentation • Commercial Grade Device
European Reliability Programs • Reliability and the Cost of Semiconductor Ownership
Reliability Testing at National Semiconductor • The Total Military/Aerospace Standardization Program
883B/RETS™ Products • MILS/RETS™ Products • 883/RETS™ Hybrids • MIL-M-38510 Class B Products
Radiation Hardened Technology • Wafer Fabrication • Semiconductor Assembly and Packaging
Semiconductor Packages • Glossary of Terms • Key Government Agencies • AN/ Numbers and Acronyms
Bibliography • MIL-M-38510 and DESC Drawing Cross Listing

TELECOMMUNICATIONS—1987

Line Card Components • Integrated Services Digital Network Components • Modems
Analog Telephone Components • Application Notes

THE SWITCHED-CAPACITOR FILTER HANDBOOK—1985

Introduction to Filters • National's Switched-Capacitor Filters • Designing with Switched-Capacitor Filters
Application Circuits • Filter Design Program • Nomographs and Tables

TRANSISTOR DATABOOK—1982

NPN Transistors • PNP Transistors • Junction Field Effect Transistors • Selection Guides • Pro Electron Series
Consumer Series • NA/NB/NR Series • Process Characteristics Double-Diffused Epitaxial Transistors
Process Characteristics Power Transistors • Process Characteristics JFETs • JFET Applications Notes

VOLTAGE REGULATOR HANDBOOK—1982

Product Selection Procedures • Heat Flow & Thermal Resistance • Selection of Commercial Heat Sink
Custom Heat Sink Design • Applications Circuits and Descriptive Information • Power Supply Design
Data Sheets

48-SERIES MICROPROCESSOR HANDBOOK—1980

The 48-Series Microcomputers • The 48-Series Single-Chip System • The 48-Series Instruction Set
Expanding the 48-Series Microcomputers • Applications for the 48-Series • Development Support
Analog I/O Components • Communications Components • Digital I/O Components • Memory Components
Peripheral Control Components

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS

ALABAMA

Arrow Electronics
1015 Henderson Rd.
Huntsville, AL 35818
(205) 837-6955

Bell Industries
1031 Putnam Drive, Suite A
Huntsville, AL 35816
(205) 837-1074

Hamilton/Avnet
4940 Research Drive NW
Huntsville, AL 35805
(205) 837-7210
TWX: 810 726 2162

Pioneer Technology
4825 University Square
Huntsville, AL 35805
(205) 837-9300
TWX: 810 726 2197

ARIZONA

Anthem Electronics
1727 E Weber Dr.
Tempe, AZ 85281
(602) 966-6600

Arrow Electronics
4134 East Wood St.
Phoenix, AZ 85040
(602) 437-0750
TWX: 910 951 1550

Bell Industries
1705 W. Fourth St.
Tempe, AZ 85281
(602) 966-7800
TWX: 910 950 0133

Hamilton/Avnet
30 South McKemy
Chandler, AZ 85226
(602) 231-5100
TWX: 66 7450

CALIFORNIA—Northern

Anthem Electronics
1040 East Brokaw Rd.
San Jose, CA 95131
(408) 295-4200
TWX: 910 338 2038

Anthem Electronics
4700 Northgate Blvd.
Suite 165
Sacramento, CA 95834
(916) 922-6800
TWX: 510 101 1419

Arrow Electronics
521 Weddell Drive
Sunnyvale, CA 94089
(408) 745-6600
TWX: 910 339 9371

Bell Industries
1161 North Fairoaks Ave.
Sunnyvale, CA 94086
(408) 734-8570
TWX: 910 339 9378

Bell Industries
500 Giuseppe Ct. Suite 6
Roseville, CA 95678
(916) 969-3100

Hamilton/Avnet
1175 Bordeaux
Sunnyvale, CA 94086
(408) 743-3355
TWX: 910 339 9332

Hamilton/Avnet
4103 Northgate Blvd.
Sacramento, CA 95834
(916) 925-2216

Time Electronics
1339 Moffet Park Dr.
Sunnyvale, CA 94089
(408) 734-9888
TWX: 172233

Zeus Components Inc., Reg 6
N CA/OR/WA/CO/
UT/NE/NM/ & AZ
1580 Old Oakland Rd. #C205
San Jose, CA 95181
(408) 998-5121

CALIFORNIA—Southern

Anthem Electronics
9369 Carroll Park Dr.
San Diego, CA 92121
(619) 453-9005
TWX: 910 335 1515

Anthem Electronics
1 Oldfield Dr.
Irvine, CA 92718
(714) 768-4444
TWX: 910 595 1583

Anthem Electronics
20640 Bahama St.
Chatsworth, CA 91311
(818) 700-1000
TWX: 910 493 2083

Arrow Electronics
9511 Ridgehaven Ct.
San Diego, CA 92123
(619) 565-4800
TWX: 910 335 1195

Arrow Electronics
2961 Dow Ave.
Tustin, CA 92680
(714) 838-5422
TWX: 910 595 2860

Arrow Electronics
19748 Dearborn St.
Chatsworth, CA 91311
(818) 701-7500
TWX: 910 493 2086

Avnet Electronics
350 McCormick Ave.
Irvine Industrial Complex
Costa Mesa, CA 92626
(714) 754-6050
TWX: 910 595 1928

Bell Industries
306 E. Alondra Blvd.
Gardena, CA 90248
(213) 515-1800

Bell Industries
12322 Monarch St.
Garden Grove, CA 92641
(714) 895-7801
TWX: 910 596 2362

Bell Industries
1829 Dehavilland Suite A
Thousand Oaks, CA 91320
(805) 499-6821
TWX: 910 321 3799

Hamilton Electro Sales
3170 Pullman Street
Costa Mesa, CA 92626
(714) 641-4159

Hamilton Electro Sales
9650 DeSoto
Chatsworth, CA 91311
(818) 700-0440

Hamilton/Avnet
1361B West 190th St.
Gardena, CA 90248
(213) 217-6751

Hamilton/Avnet
4545 Viewridge Avenue
San Diego, CA 92123
(619) 571-7510
TWX: 695 415

Hamilton/Avnet
3002 East G Street
Ontario, CA 91764
(714) 989-4602

Time Electronics
370 South Crenshaw Blvd.
Suite E-104
Torrance, CA 90503-1727
(213) 320-0880
TWX: 910 349 6650

Time Electronics
2410 E. Cerritos Ave.
Anaheim, CA 92806
(714) 934-0911
TWX: 910 591 1234

Time Electronics
8525 Arjons Drive
San Diego, CA 92126
(619) 586-1331
TWX: 858902

Time Electronics
9751 Independence Ave.
Chatsworth, CA 91311
(818) 998-7200
TWX: 910 380 6274

Zeus Components Inc., Reg 5
San Fernando Valley
5236 Colodny Drive
Agoura Hills, CA 91301
(818) 889-3838

Zeus Components Inc., Reg 5H
All Hughes
22700 Savy Ranch Pkwy.
Yorba Linda, CA 92686

Zeus Components Inc., Reg 8
S CA, SG VLY, OC, SD CTY
22700 Savy Ranch Pkwy.
Yorba Linda, CA 92686
(714) 921-9000

COLORADO

Anthem Electronics
373 Inverness Dr. South
Englewood, CO 80112
(303) 790-4500

Arrow Electronics
7060 S Tucson Way
Suite 136
Englewood, CO 80112
(303) 790-4444
TWX: 910 931 2626

Bell Industries
8155 W. 48th Avenue
Wheatridge, CO 80033
(303) 424-1985
TWX: 910 938 0393

Hamilton/Avnet
8765 E Orchard Road #708
Englewood, CO 80111
(303) 779-9998
TWX: 910 935 0787

CONNECTICUT

Arrow Electronics
12 Beaumont Rd.
Wallingford, CT 06492
(203) 265-7741
TWX: 710 476 0162

Hamilton/Avnet
Commerce Drive
Commerce Park
Danbury, CT 06810
(203) 797-2800

Anthem Electronics
170 Research Parkway
Meridan, CT 06450
(203) 237-2282

Pioneer Northeast
112 Main St.
Norwalk, CT 06852
(203) 853-1515
TWX: 710 468 3378

Time Electronics
1701 Highland Ave.
Cheshire, CT 06410
(203) 271-3200
TWX: 910 380 6270

FLORIDA

Arrow Electronics
400 Fairway Drive
Deerfield Beach, FL 33441
(305) 429-8200
TWX: 510 955 9456

Arrow Electronics
37 Skyline Drive #3101
Lake Mary, FL 32746
(407) 323-0252

Bell Industries
10810 72nd St. North #201
Suite 201
Largo, FL 33543
(813) 541-4434

Bell Industries
638 South Military Trail
Deerfield Beach, FL 33442
(305) 421-1997

Hamilton/Avnet
6801 N.W. 15th Way
Ft. Lauderdale, FL 33309
(305) 971-2900
TWX: 510 956 3097

Hamilton/Avnet
3197 Tech Drive North
St. Petersburg, FL 33702
(813) 576-3930
TWX: 810 863 0374

Hamilton/Avnet
6947 University Blvd.
Winter Park, FL 32792
(305) 628-3888

Pioneer Technology
221 North Lake Blvd.
Altamonte Springs, FL 32701
(305) 834-9090
TWX: 810 853 0284

Pioneer Technology
674 South Military Trial
Deerfield Beach, FL 33441
(305) 428-8877
TWX: 510 955 9653

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS (Continued)

FLORIDA (Continued)

Zeus Components Inc., Reg 4
FL, GA, AL, MI, SC & TN
1750 West Broadway
Oviedo, FL 32765
(305) 365-3000

GEORGIA

Arrow Electronics
3155 Northwoods Parkway
Suite A
Norcross, GA 30071
(404) 449-8252
TWX: 810 766 0439

Bell Industries
6690-C Jones Mill Ct.
Norcross, GA 30092
(404) 662-0923

Hamilton/Avnet
5825D Peach Tree Corner E
Norcross, GA 30092
(404) 447-7500

Pioneer Technology
3100F Northwoods Place
Norcross, GA 30071
(404) 448-1711
TWX: 810 766 4515

ILLINOIS

Anthem Electronics
180 Crossen Ave.
Elk Grove Village, IL 60007
(312) 640-6066

Arrow Electronics
1140 West Thorndale Avenue
Itasca, IL 60143
(312) 250-0500
TWX: 910 222 0351

Bell Industries
515 Busse Road
Elk Grove Village, Il 60007
(312) 640-1910
TWX: 910 223 4519

Bell Industries
730 West Kilarney
Urbana, IL 61801
(217) 328-1077

Hamilton/Avnet
1130 Thorndale Ave.
Bensenville, IL 60106
(312) 860-7780

Pioneer Electronics
2171 Executive Dr., Suite 200
Addison, IL 60101
(312) 495-9680
TWX: 910 222 1834

INDIANA

Advent Electronics Inc.
8446 Moller Rd.
Indianapolis, IN 46268
(317) 872-4910
TWX: 810 341 3228

Arrow Electronics
2495 Directors Row
Suite H
Indianapolis, IN 46241
(317) 243-9353
TWX: 810 341 3119

Bell Industries
3606 E. Maumee Ave.
Fort Wayne, IN 46803
(219) 423-3422
TWX: 910 997 0701

Bell Industries—Graham Div.
133 S Pennsylvania St.
Indianapolis, IN 46204
(317) 834-8202
TWX: 810 341 3481

Hamilton/Avnet
485 Gradle Dr.
Carmel, IN 46032
(317) 844-9333
TWX: 810 260 3966

Pioneer-Indiana
6408 Castleplace Drive
Indianapolis, IN 46250
(317) 849-7300

IOWA

Advent Electronics
682 58th Ave. Court S.W.
Cedar Rapids, IA 52404
(319) 363-0221
TWX: 910 525 1337

Arrow Electronics
375 Collins Rd. N.E.
Cedar Rapids, IA 52402
(319) 395-7230
TWX: 910 493 2086

Bell Industries
1221 Park Place N.E.
Cedar Rapids, IA 52402
(319) 395-0730

Hamilton/Avnet
915 33rd Avenue S.W.
Cedar Rapids, IA 52404
(319) 362-4757

KANSAS

Arrow Electronics
8208 Melrose Dr.
Suite 210
Lenexa, KS 66214
(913) 541-9542

Hamilton/Avnet
9219 Quivira Rd
Overland Park, KS 66215
(913) 888-8900

Pioneer Standard
10551 Lackmann Road
Lenexa, KS 66215
(913) 492-0500

MARYLAND

Arrow Electronics
8300 Guilford Dr.
Columbia, MD 21045
(301) 995-0003
TWX: 710 236 9005

Hamilton/Avnet
6822 Oak Hall Lane
Columbia, MD 21045
(301) 995-3500
TWX: 710 862 1861

Anthem Electronics
9020-A Mendenhall Court
Columbia, MD 21045
(301) 964-0040
TWX: 710 862 1909

Pioneer Technology
9100 Gaither Road
Gaithersburg, MD 20877
(301) 921-0660
TWX: 710 828 0545

Time Electronics
9051 Red Branch Rd.
Columbia, MD 21045
(301) 964-3090
TWX: 710 862 2860

Zeus Components Inc., Reg 2
MD, DE, VA, WVA,
NC, RAYTHEON
8930 Route 108
Columbia, MD 21045
(301) 997-1118

MASSACHUSETTS

Arrow Electronics
25 Upton Drive
Wilmington, MA 01887
(617) 935-5134
TWX: 710 393 6770

Gerber Electronics
128 Carnegie Row
Norwood, MA 02062
(617) 769-6000
TWX: 710 336 1987

Hamilton/Avnet
10D Centennial Dr.
Peabody, MA 01960
(617) 531-7430
TWX: 710 393 0382

Anthem Electronics
38 Jonspin Road
Wilmington, MA 01887
(617) 657-5170
TWX: 710 332 1387

Pioneer Northeast
44 Hartwell Avenue
Lexington, MA 02173
(617) 861-9200
TWX: 710 326 6617

Time Electronics
10 A Centennial Drive
Peabody, MA 01960
(617) 532-6200
TWX: 710 393 0171

Zeus Coomponents Inc., Reg 1A
MA, RI, VT, NH, ME & CANADA
429 Marrett Rd.
Lexington, MA 02173
(617) 863-8800

MICHIGAN

Arrow Electronics
3510 Roger Chaffee
Memorial Blvd. S.E.
Grand Rapids, MI 49508
(616) 243-0912

Arrow Electronics
755 Phoenix Dr.
Ann Arbor, MI 48108
(313) 971-8220
TWX: 810 223 6020

Bell Industries
814 Phoenix Dr.
Ann Arbor, MI 48104
(313) 971-9093

Hamilton/Avnet
2215 29th St. S.E.
Grand Rapids, MI 49508
(616) 243-8805
TWX: 810 273 6921

Hamilton/Avnet
32487 Schoolcraft Road
Livonia, MI 48150
(313) 522-4700

Pioneer Standard
4505 Broadmoor S.E.
Grand Rapids, MI 49508
(616) 698-1800
TWX: 510 600 8456

Pioneer Standard
13485 Stanford
Livonia, MI 48150
(313) 525-1800
TWX: 810 242 3271

R. M. Electronics
4310 Roger B Chaffee
Wyoming, MI 49508
(616) 531-9300

MINNESOTA

Anthem Electronics
10025 Valley View Rd. #160
Eden Prairie, MN 55344
(612) 944-5454

Arrow Electronics
5230 73rd Street
Edina, MN 55435
(612) 830-1800
TWX: 910 576 3125

Hamilton/Avnet
12400 Whitewater Dr.
Minnetonka, MN 55343-9421
(612) 932-0600
TWX: 910 572 2867

Pioneer-Twin Cities
7625 Golden Triangle Dr.
Suite G
Eden Prairie, MN 55344
(612) 935-5444
TWx: 910 576 2738

MISSOURI

Arrow Electronics
2380 Schuetz Road
St. Louis, MO 63146
(314) 567-6888
TWX: 910 764 0882

Hamilton/Avnet
13743 Shoreline Ct.-East
Earth City, MO 83045
(314) 344-1200
TWX: 910 762 0627

Time Electronics
330 Sovereign Ct.
St. Louis, MO 63011-4491
(314) 391-6444
TWX: 910 760 1893

NEW HAMPSHIRE

Arrow Electronics
3 Perimeter Rd.
Manchester, NH 03103
(603) 668-6968
TWX: 710 220 1684

Bell Industries—C & H Div.
19 Park Avenue
Hudson, NH 03051
(603) 882-1133
TWX: 710 228 8959

Hamilton/Avnet
444 Industrial Dr.
Manchester, NH 03102
(603) 624-9400

NEW JERSEY—Northern

Arrow Electronics
6 Century Drive
Parsippany, NJ 07054
(201) 575-5300
TWX: 710 734 4403

Hamilton/Avnet
10 Industrial Rd.
Fairfield, NJ 07006
(201) 575-3390
TWX: 710 734 4409

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS (Continued)

NEW JERSEY—Northern (Continued)

Anthem Electronics
311 Rt. 46 West
Fairfield, NJ 07006
(201) 227-7960
TWX: 710 734 4312

Nu Horizons Electronics
39 Route 46
Pinebrook, NJ 07058
(201) 882-8300

Pioneer
45 Route 46
Pine Brook, NJ 07058
(201) 575-3510
TWX: 710 734 4382

NEW JERESY—Southern

Arrow Electronics
6000 Lincoln Drive East
Marlton, NJ 08053
(609) 596-8000
TWX: 710 897 0829

Hamilton/Avnet
One Keystone Ave.
Cherry Hill, NJ 08003
(609) 424-0100
TWX: 710 940 0262

NEW MEXICO

Alliance Electronics Inc.
11030 Cochiti S.E.
Albuquerque, NM 87123
(505) 292-3360
TWX: 910 989 1151

Arrow Electronics
2460 Alamo Ave. S.E.
Albuquerque, NM 87106
(505) 243-4566
TWX: 910 989 1679

Bell Industries
11728 Linn N.E.
Albuquerque, NM 87123
(505) 292-2700
TWX: 910 989 0625

Hamilton/Avnet
2524 Baylor Drive S.E.
Albuquerque, NM 87106
(505) 765-1500
TWX: 910 989 1631

NEW YORK—Upstate

Arrow Electronics
3375 Brighton-Henrietta
Townline Rd.
Rochester, NY 14623
(716) 427-0300
TWX: 510 253 4766

Hamilton/Avnet
103 Twin Oaks Drive
Syracuse, NY 13206
(315) 437-2641
TWX: 710 541 1506

Hamilton/Avnet
2060 Town Line Road
Rochester, NY 14623
(716) 475-9130
TWX: 510 253 5470

Pioneer Northeast
68 Corporate Drive
Binghamton, NY 13904
(607) 722-9300
TWX: 510 252 0893

Pioneer Northeast
840 Fairport Rd.
Fairport, NY 14450
(716) 381-7070
TWX: 510 253 7001

Summit Electronics
916 Main Street
Buffalo, NY 14202
(716) 887-2800
TWX: 710 522 1692

Summit Electronics
292 Commerce Drive
Rochester, NY 14623
(716) 334-8110

Time Electronics
6075 Corporate Dr.
East Syracuse, NY 13057
(315) 432-0355
TWX: 510 100 6192

NEW YORK—Metro Area

Arrow Electronics
20 Oser Ave.
Hauppauge, NY 11788
(516) 231-1000
TWX: 510 227 6623

Hamilton/Avnet
833 Motor Parkway
Hauppauge, NY 11788
(516) 434-7413

Hamilton/Avnet Export Div.
1065 Old Country Rd., #211A
Westbury, NY 11590
(516) 997-6868

Anthem Electronics
400 Oser Ave.
Hauppauge, NY 11787
(516) 273-1660
TWX: 510 227 1042

Nu Horizons Electronics
6000 New Horizons Blvd.
Amityville, NY 11701
(516) 226-6000

Pioneer
60 Crossways Park West
Woodbury, NY 11797
(516) 921-8700
TWX: 710 326 6617

Time Electronics
70 Marcus Boulevard
Hauppauge, NY 11788
(515) 273-0100
TWX: 858881

Zeus Components Inc., Reg 1
NY/ROCK/NJ/E PA/ CT
100 Midland Ave.
Port Chester, NY 10573
(914) 937-7400

Zeus Components Inc., Reg 1B
Long Island/NYC
2110 Smithtown Ave.
Ronkonkoma, NY 11779
(516) 737-4500

NORTH CAROLINA

Arrow Electronics
5240 Greens Dairy Rd.
Raleigh, NC 27604
(919) 876-3132
TWX: 510 928 1856

Arrow Electronics
938 Burke Street
Winston-Salem, NC 27101
(919) 725-8711
TWX: 510 931 3169

Hamilton/Avnet
3510 Spring Forest Road
Raleigh, NC 27601
(919) 878-0810
TWX: 510 928 1836

Pioneer Technology
9801-A Southern Pine Blvd.
Charlotte, NC 28210
(704) 527-8188
TWX: 810 621 0366

OHIO

Arrow Electronics
7620 McEwen Rd.
Centerville, OH 45459
(513) 435-5563
TWX: 810 459 1611

Arrow Electronics
6238 Cochran Rd.
Solon, OH 44139
(216) 248-3990
TWX: 810 427 9409

Bell Industries
444 Windsor Park Drive
Dayton, OH 45459
(513) 435-8660

Bell Industries
Micro-Mil Division
118 Westpark Road
Dayton, OH 45459
(513) 434-8231
TWX: 810 459 1615

CAM/OHIO Electronics
749 Miner Road
Highland Heights, OH 44143
(216) 461-4700
TWx: 810 427 2976

Hamilton/Avnet
954 Senate Drive
Dayton, OH 45459
(513) 439-6700
TWX: 810 450 2531

Hamilton/Avnet
30325 Bainbridge Rd., Bldg. A
Solon, OH 44139
(216) 831-3500
TWX: 810 427 9452

Hamilton/Avnet
777 Brooksedge Blvd.
Westerville, OH 43081
(614) 882-7004

Pioneer Standard
4800 East 131st Street
Cleveland, OH 44105
(216) 587-3600
TWX: 810 422 2210

Pioneer Standard
4433 Interpoint Blvd.
Dayton, OH 45424
(513) 236-9900
TWX: 810 459 1683

Zeus Components Inc., Reg 3
Dayton (DESC)
2912 Springboro St., Ste. 106
Dayton, OH 45439
(914) 937-7400

OKLAHOMA

Arrow Electronics
12111 E. 51st Street
Tulsa, OK 74146
(918) 252-7537

Hamilton/Avnet
12121 East 51st St.
Suite 102A
Tulsa, OK 74146
(918) 252-7297

Quality Components
3158 South 108th East Ave.
Suite 274
Tulsa, OK 74146
(918) 664-8812

Radio Inc.
1000 South Main Street
Tulsa, OK 74119
(918) 587-9123
TWX: 49 2429

OREGON

Almac-Stroum Electronics
1885 N.W. 169th Place
Beaverton, OR 97006
(503) 629-8090
TWX: 910 467 8743

Anthem Electronics
9705 S.W. Sunshine Ct.
Suite 900
Beaverton, OR 97005
(503) 643-1114

Arrow Electronics
1800 N.W 167th Place
Suite 145
Beaverton, OR 97006
(503) 645-6456
TWX: 910 464 0007

Bell Industries
6024 S.W. Jean Rd.
Lake Oswego, OR 97034
(503) 241-4115
TWX: 910 455 8177

Hamilton/Avnet
6024 S.W. Jean Rd.
Bldg. C, Suite 10
Lake Oswego, OR 97034
(503) 635-7850

PENNSYLVANIA—Eastern

Arrow Electronics
650 Seco Rd.
Monroeville, PA 15146
(412) 856-7000
TWX: 710 797 3894

CAM/RPC IND Electronics
620 Alpha Drive
RIDC Park
Pittsburgh, PA 15238
(412) 782-3770
TWX: 710 795 3126

Hamilton/Avnet
2800 Liberty Ave.
Pittsburgh, PA 15227
(412) 281-4150

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS (Continued)

PENNSYLVANIA—Eastern (Continued)

Anthem Electronics
101 Rock Road
Horsham, PA 19044
(215) 443-5150

Pioneer Technology
261 Gibraltar Road
Horsham, PA 19044
(215) 674-4000
TWX: 510 665 6778

Pioneer-Pittsburgh
259 Kappa Drive
Ridgepark
Pittsburgh, PA 15238
(412) 782-2300
TWX: 710 795 3122

Time Electronics
600 Clark Ave.
King of Prussia, PA 19406
(215) 337-0900
TWX: 845317

TENNESSEE

Bell Industries
Instate: (800) 752-2050
Out of State: (800) 433-8301

TEXAS

Arrow Electronics
3220 Commander Dr.
Carrollton, TX 75006
(214) 380-6464
TWX: 910 860 5377

Arrow Electronics
2227 West Braker Lane
Austin, TX 78758
(512) 835-4180
TWX: 910 874 1348

Arrow Electronics
10899 Kinghurst Dr.
Suite 100
Houston, TX 77099
(713) 530-4700
TWX: 910 880 439

Hamilton/Avnet
2111 West Walnut Hill Ln.
Irving, TX 75062
(214) 550-7755
TWX: 07 32359

Hamilton/Avnet
4850 Wright Road #190
Staford, TX 77477
(713) 240-7733
TWX: 910 881 5523

Hamilton/Avnet
1807A West Braker Lane
Austin, TX 78758
(512) 837-8911
TWX: 910 874 1319

Pioneer Electronics
1826 Kramer Lane
Suite D
Austin, TX 78758
(512) 835-4000

Pioneer Standard
13710 Omega Road
Dallas, TX 75240
(214) 386-7300
TWX: 910 860 5563

Pioneer Electronics
5853 Point West Drive
Houston, TX 77036
(713) 988-5555
TWX: 910 881 1606

Quality Components
1005 Industrial Blvd.
Sugarland, TX 77478
(713) 240-2255
TWX: 910 881 7251

Quality Components
2120M Braker Lane
Austin, TX 78758
(512) 835-0220
TWX: 910 874 1377

Quality Components
4257 Kellway Circle
Addison, TX 75001
(214) 733-4300
TWX: 910 660 5459

Zeus Components Inc., Reg 7
TX, AR, OK, LA, KS, MO, IO, NE
1800 N. Glenville Rd.
Richardson, TX 75081
(214) 783-7010

UTAH

Anthem Electronics
1615 West 2200 South #A
Salt Lake City, UT 84119
(801) 973-8555

Arrow Electronics
1946 W. Parkway Blvd.
Salt Lake City, UT 84119
(801) 973-6913

Bell Industries
3639 West 2150 South
Salt Lake City, UT 84120
(801) 972-6969

Hamilton/Avnet
1585 West 2100 South
Salt Lake City, UT 84117
(801) 972-4300
TWX: 910 925 4018

WASHINGTON

Almac-Stroum Electronics
14360 S.E. Eastgate Way
Bellevue, WA 98007
(206) 643-9992
TWX: 910 444 2067

Anthem Electronics
5020 148th Ave. N.E.
Suite 103
Redmond, WA 98052
(206) 881-0850
TWX: 910 997 0118

Arrow Electronics
19450 68th Ave. South
Kent, WA 98032
(206) 575-4420
TWX: 910 444 2034

Hamilton/Avnet
14212 North East 21st
Bellevue, WA 98005
(206) 453-5844

WISCONSIN

Arrow Electronics
200 N. Patrick Blvd.
Brookfield, WI 53005
(414) 792-0150
TWX: 910 262 1193

Bell Industries
W227 N913 Westmound Ave.
Waukesha, WI 53186
(414) 547-8879
TWX: 910 262 1156

Hamilton/Avnet
2975 Moorland Rd.
New Berlin, WI 53151
(414) 784-4516

Taylor Electric
1000 West Donges Bay Road
Mequon, WI 53092
(414) 241-4321
TWX: 910 262 3414

CANADA

Electro Sonic Inc.
1100 Gordon Baker Road
Willowdale, Ontario M2H 3B3
(416) 494-1666
TWX: 06 525295

Hamilton/Avnet
2550 Boundary Rd. #105
Burnaby, B.C., V5M 3Z0
(604) 437-6667

Hamilton/Avnet
2816 21st N.E.
Calgary, Alberta T2E 6Z2
(403) 250-9380
TWX: 03 827642

Hamilton/Avnet
2795 Rue Halpern
St. Laurent, Quebec H4S 1P8
(514) 335-1000
TWX: 610 421 3731

Hamilton/Avnet
6845 Redwood Drive 3, 4, 5
Mississauga, Ontario L4V 1T1
(416) 677-7432
TWX: 610 492 8867

Hamilton/Avnet
190 Colonnade Rd.
Nepean, Ontario K2E 7L5
(613) 226-1700
TWX: 053 4971

Semad Electronics Ltd.
243 Place Frontenac
Pointe Claire, Quebec H9R 4Z7
(514) 694-0860

Semad Electronics Ltd.
3700 Gilmore Way #210
Burnaby, B.C. V5G 4M1
(604) 438-2515

Semad Electronics Ltd.
75 Glendeer Dr. S.E. #210
Calgary, Alberta T2H 2S8
(403) 252-5664

Semad Electronics Ltd.
1827 Woodward Dr. #303
Ottawa, Ontario K2C 0R3
(613) 727-8325

Zentronics
8 Tilbury Ct.
Brampton, Ontario L6T 3T4
(416) 451-9600
TWX: 06 97678

Zentronics
Edmonton Sales Office
Edmonton, Alberta T6N 1B2
(403) 468-8306

Zentronics
11400 Bridgeport Rd., Unit 108
Richmond, B.C. V6X 1T2
(604) 273-5575
TWX: 04 355844

Zentronics
155 Colonade Rd. So.
Units 17 & 18
Nepean, Ontario K2E 7K1
(613) 226-8840

Zentronics
817 McCaffrey St.
Ville St. Laurent, Quebec H4T 1N3
(514) 737-9700

Zentronics
93-1313 Border St.
Winnipeg, Manitoba R3H 0X4
(204) 694-1957

Zentronics
Waterloo Sales Office
Waterloo, Quebec H4T 1N3
(800) 387-2329

Zentronics
Saskatoon Sales Office
Saskatoon, Alberta R3H 0X4
(306) 955-2207

Zentronics
6815 8th St. N.E.
Suite 100
Calgary, Alberta T2E 7H7
(403) 272-1021
TWX: 04 355844

SALES OFFICES

ALABAMA
Huntsville
(205) 837-8960
(205) 721-9367

ARIZONA
Tempe
(602) 966-4563

B.C.
Burnaby
(604) 435-8107

CALIFORNIA
Encino
(818) 888-2602
Inglewood
(213) 645-4226
Roseville
(916) 786-5577
San Diego
(619) 587-0666
Santa Clara
(408) 562-5900
Tustin
(714) 259-8880
Woodland Hills
(818) 888-2602

COLORADO
Boulder
(303) 440-3400
Colorado Springs
(303) 578-3319
Englewood
(303) 790-8090

CONNECTICUT
Fairfield
(203) 371-0181
Hamden
(203) 288-1560

FLORIDA
Boca Raton
(305) 997-8133
Orlando
(305) 629-1720
St. Petersburg
(813) 577-1380

GEORGIA
Atlanta
(404) 396-4048
Norcross
(404) 441-2740

ILLINOIS
Schaumburg
(312) 397-8777

INDIANA
Carmel
(317) 843-7160
Fort Wayne
(219) 484-0722

IOWA
Cedar Rapids
(319) 395-0090

KANSAS
Overland Park
(913) 451-8374

MARYLAND
Hanover
(301) 796-8900

MASSACHUSETTS
Burlington
(617) 273-3170
Waltham
(617) 890-4000

MICHIGAN
W. Bloomfield
(313) 855-0166

MINNESOTA
Bloomington
(612) 835-3322
(612) 854-8200

NEW JERSEY
Paramus
(201) 599-0955

NEW MEXICO
Albuquerque
(505) 884-5601

NEW YORK
Endicott
(607) 757-0200
Fairport
(716) 425-1358
(716) 223-7700
Melville
(516) 351-1000
Wappinger Falls
(914) 298-0680

NORTH CAROLINA
Cary
(919) 481-4311

OHIO
Dayton
(513) 435-6886
Highland Heights
(216) 442-1555
(216) 461-0191

ONTARIO
Mississauga
(416) 678-2920
Nepean
(404) 441-2740
(613) 596-0411
Woodbridge
(416) 746-7120

OREGON
Portland
(503) 639-5442

PENNSYLVANIA
Horsham
(215) 675-6111
Willow Grove
(215) 657-2711

PUERTO RICO
Rio Piedias
(809) 758-9211

QUEBEC
Dollard Des Ormeaux
(514) 683-0683
Lachine
(514) 636-8525

TEXAS
Austin
(512) 346-3990
Houston
(713) 771-3547
Richardson
(214) 234-3811

UTAH
Salt Lake City
(801) 322-4747

WASHINGTON
Bellevue
(206) 453-9944

WISCONSIN
Brookfield
(414) 782-1818
Milwaukee
(414) 527-3800

National Semiconductor

National Semiconductor
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-5000
TWX: (910) 339-9240

SALES OFFICES (Continued)

INTERNATIONAL OFFICES

Electronica NSC de Mexico SA
Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: 52-5-524-9402

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 1409
6 Andor Salas 62/64
01451 Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Telex: 391-1131931 NSBR BR

National Semiconductor GmbH
Industriestrasse 10
D-8080 Furstenfeldbruck
West Germany
Tel: 49-08141-103-0
Telex: 527 649

National Semiconductor (UK) Ltd.
301 Harpur Centre
Horne Lane
Bedford MK40 ITR
United Kingdom
Tel: (02 34) 27 00 27
Telex: 826 209

National Semiconductor Benelux
Vorstlaan 100
B-1170 Brussels
Belgium
Tel: (02) 6725360
Telex: 61007

National Semiconductor (UK) Ltd.
1, Bianco Lunos Alle
DK-1868 Fredriksberg C
Denmark
Tel: (01) 213211
Telex: 15179

National Semiconductor
Expansion 10000
28, rue de la Redoute
F-92260 Fontenay-aux-Roses
France
Tel: (01) 46 60 81 40
Telex: 250956

National Semiconductor S.p.A.
Strada 7, Palazzo R/3
20089 Rozzano
Milanofiori
Italy
Tel: (02) 8242046/7/8/9

National Semiconductor AB
Box 2016
Stensatravagen 13
S-12702 Skarholmen
Sweden
Tel: (08) 970190
Telex: 10731

National Semiconductor
Calle Agustin de Foxa, 27
28036 Madrid
Spain
Tel: (01) 733-2958
Telex: 46133

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
Ch-8304 Wallisellen-Zurich
Switzerland
Tel: (01) 830-2727
Telex: 59000

National Semiconductor
Kauppakartanonkatu 7
SF-00930 Helsinki
Finland
Tel: (0) 33 80 33
Telex: 126116

National Semiconductor Japan Ltd.
Sanseido Bldg. 5F
4-15 Nishi Shinjuku
Shinjuku-ku
Tokyo 160 Japan
Tel: 3-299-7001
Fax: 3-299-7000

National Semiconductor Hong Kong Ltd.
Southeast Asia Marketing
Austin Tower, 4th Floor
22-26A Austin Avenue
Tsimshatsui, Kowloon, H.K.
Tel: 852 3-7243645
Cable: NSSEAMKTG
Telex: 52996 NSSEA HX

National Semiconductor (Australia) PTY, Ltd.
1st Floor, 441 St. Kilda Rd.
Melbourne, 3004
Victory, Australia
Tel: (03) 267-5000
Fax: 61-3-2677458

National Semiconductor (PTE), Ltd.
200 Cantonment Road 13-01
Southpoint
Singapore 0208
Tel: 2252226
Telex: RS 33877

National Semiconductor (Far East) Ltd.
Taiwan Branch
P.O. Box 68-332 Taipei
7th Floor, Nan Shan Life Bldg.
302 Min Chuan East Road,
Taipei, Taiwan R.O.C.
Tel: (86) 02-501-7227
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (Far East) Ltd.
Korea Office
Room 612,
Korea Fed. of Small Bus. Bldg.
16-2, Yoido-Dong,
Youngdeungpo-Ku
Seoul, Korea
Tel: (02) 784-8051/3 – 785-0696-8
Telex: K24942 NSRKLO

 TL2376P RRD-RRD90M078/Printed in U.S.A.